REAL-TIME CONTROL OVER NETWORKS

A Dissertation

by

KUN JI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2006

Major Subject: Mechanical Engineering

REAL-TIME CONTROL OVER NETWORKS

A Dissertation

by

KUN JI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Won-jong Kim
Committee Members,	Alan Palazzolo
	Darbha Swaroop
	Yoonsuck Choe
Head of Department,	Dennis O'Neal

May 2006

Major Subject: Mechanical Engineering

ABSTRACT

Real-Time Control Over Networks.

(May 2006)

Kun Ji,

B. S., Tsinghua University, Beijing, China;

M. S., Tsinghua University, Beijing, China

Chair of Advisory Committee: Dr. Won-jong Kim

A control system in which sensors, actuators, and controllers are interconnected over a communication network is called a networked control system (NCS). Enhanced computational capabilities and bandwidths in the networking technology enabled researchers to develop NCSs to implement distributed control schemes. This dissertation presents a framework for the modeling, design, stability analysis, control, and bandwidth allocation of real-time control over networks. This framework covers key research issues regarding control over networks and can be the guidelines of NCS design. A single actuator ball magnetic-levitation (maglev) system is implemented as a test bed for the real-time control over networks to illustrate and verify the theoretical results of this dissertation. Experimentally verifying the feasibility of Internet-based real-time control is another main objective of this dissertation.

First, this dissertation proposes a novel NCS model in which the effects of the network-induced time delay, data-packet loss, and out-of-order data transmission are all considered. Second, two simple algorithms based on model-estimator and predictor- and timeout-scheme are proposed to compensate for the network-induced time delay and packet loss simultaneously. These algorithms are verified experimentally by the ball maglev test bed. System stability

analyses of original and compensated systems are presented. Then, a novel co-design consideration related to real-time control and network communication is also proposed. The working range of the sampling frequency is determined by the analysis of the system stability and network parameters such as time delay, data rate, and data-packet size. The NCS design chart developed in this dissertation can be a useful guideline for choosing the network and control parameters in the design of an NCS. Using a real-time operating system for real-time control over networks is also proposed as one of the main contributions of this dissertation.

After a real-time NCS is successfully implemented, advanced control theories such as robust control, optimal control, and adaptive control are applied and formulated to improve the quality of control (QoC) of NCSs. Finally, an optimal dynamic bandwidth management method is proposed to solve the optimal network scheduling and bandwidth allocation problem when NCSs are connected to the same network and are sharing the network resource.

To my parents

and

wife Jinqiao Wang

ACKNOWLEDGMENTS

I would like to express deep gratitude to my dissertation advisor Dr. Won-jong Kim for his consistent encouragement, motivation, and moral as well as financial support throughout my study at Texas A&M University. I appreciate his giving me excellent opportunities to publish and present my work several times and get recognition for it. My interaction with him over the past 3 years was a great learning experience in all aspects of research.

I wish to thank Drs. Palazzolo, Swaroop, and Choe for serving as my committee members. I sincerely appreciate the time, advice, and support they offered me during my doctoral study. The significant contributions of the committee members brought this dissertation to success.

I would like to thank Steve Paschall, a former undergraduate student of Dr. Kim, for design and implementation of the ball Maglev system. Then I would like to give my special thanks to Abhinav Srivastava and Ajit Ambike, former Master's students of Dr. Kim, for their original contributions in Internet-based supervisory control and real-time operating environment implementation. I appreciate all my labmates for the friendly and enjoyable atmosphere in the lab.

My special appreciation goes to our research sponsor. This material is based upon work supported by National Science Foundation under Grant No. CMS-0116642 through Dr. Won-jong Kim.

Finally, I would like to thank my wife for her love and support over the years. She always urged me to go on when my patience ran short and confidence low. I could never thank my parents enough for their unconditional love and prayers, without which I would never have come so far.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION *

1.1 Computer Controlled Systems

With the development of computer technologies, computer controlled systems became more and more popular. Computer controlled systems can be roughly classified into three modes: 1) direct digital control system (DDCS), 2) distributed control system (DCS), and 3) networked control system (NCS). There is a significant trend in modern automation industry to perform complex remote operations by integrating computing, network communication, and real-time control. Real-time control over networks became possible and caught many research attentions.

1.1.1 Direct Digital Control Systems

The name DDCS emphasizes that the computer controls the process directly. At the beginning stage of computer development, the use of digital computers as control system components was limited because the computers were too bulky, not highly reliable, and consumed too much power. Early computers in control systems were operated only as either operator-guide or set-point control. Analog controllers were still needed. This situation changed when more small and reliable digital computers became available with the fast development of computer technology.

The block diagram of a DDCS setup is shown in Fig. 1-1. In a DDCS, the analog

---

This dissertation follows format of *IEEE Transactions on Automatic Control*.

controller is replaced by a digital computer. Sensors and actuators are connected to the digital computer with point-to-point connections. Real-time control tasks such as sensing, control signal calculation, and actuation are all performed by the digital computer.

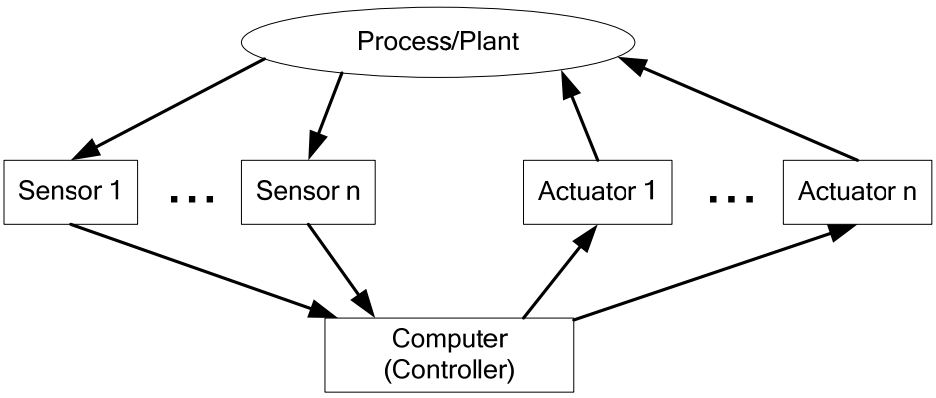


Fig. 1-1. Block diagram of a DDCS

1.1.2 Distributed Control Systems

The DCS also arose with the development of computer technology and the increase of the scale of the control system. In a DCS shown in Fig. 1-2, distribution of computation power is required and several interacting computers are connected to a network and sharing the same workload. A DCS generally consists of process stations where the process is controlled and operator stations where the process is monitored by the operators. Most of the real-time control tasks (sensing, control calculation, and actuation) are carried out within the individual process stations themselves, i.e. the control loops are all closed locally. Messages such as monitoring information, on/off command signals, alarm information, and so on are transmitted over the network. The applications of a DCS can be found in tele-operation shown in Fig. 1-3 and

supervisory control shown in Fig. 1-4. The network is used for sending instantaneous feedback for monitoring and user commands for corrective actions in the event of emergency.

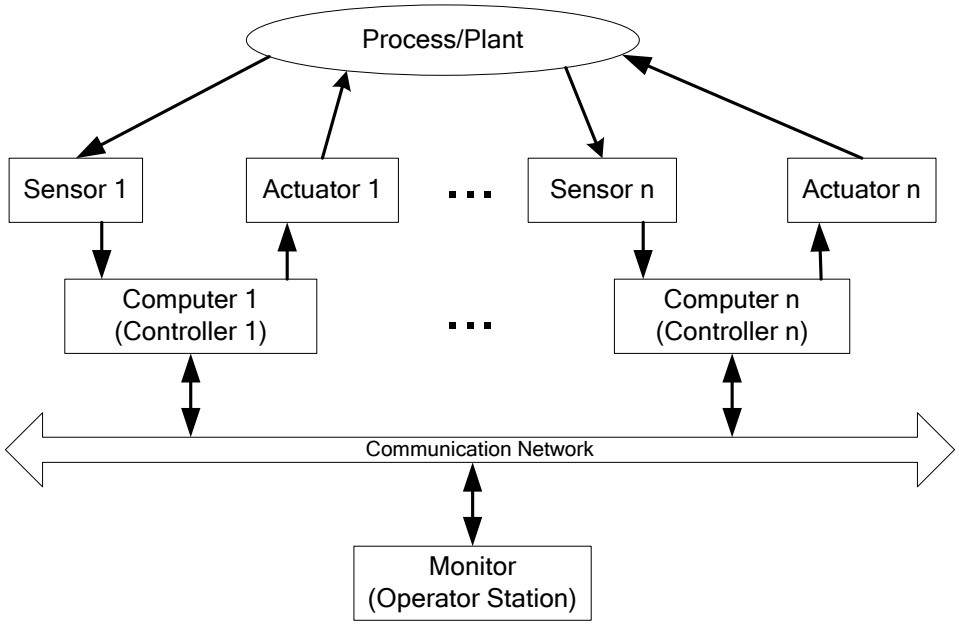


Fig. 1-2. Block diagram of a DCS

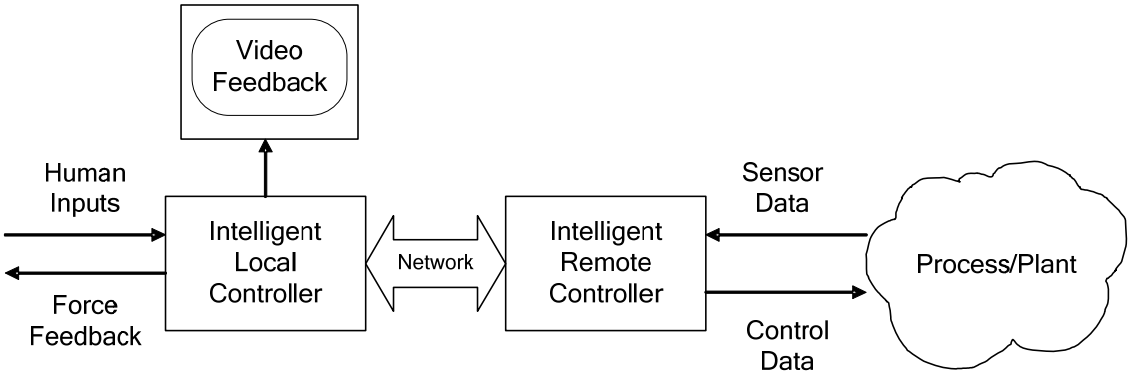


Fig. 1-3. Block diagram of a tele-operated system

Tele-operation is an elementary application of a DCS. In tele-operation, a human operator conducts a task using a remote manipulator to perform various tasks in hazardous environments like space, underwater, or an atomic power plant. Here are key characteristics of a tele-operated system in the context of a general factory operation. First, the time delay, limited signal bandwidth, and signal noise, etc. hinder the smooth communication between the operator and the process/plant. The performance of the operator is affected significantly by those factors. Second, there are uncertainties that the operator has to deal with. For example, if the environment is not accurately modeled, there is difference between the actual environment and the model, and the user has to use his perception to deal with such differences.

Tele-operations based on master-slave systems have been in use since 1960's [1–2]. Yokokohji and Yoshikawa studied the design of master-slave systems for superior maneuverability of robot manipulator [3]. A one-degree-of-freedom (DOF) system including the operator and object dynamics was analyzed, and a control scheme was proposed to provide ideal kinesthetic coupling. Tele-operation is also used in the field of surgery. The Black Falcon developed by Madhani *et al*. implemented a tele-operated surgical instrument for minimally invasive surgery (MIS) [4]. MIS is a practice of performing surgery through small incisions using specialized surgical instruments. To feel the instrument-tissue interactions, the surgeon is given a force feedback. Mitsuishi *et al*. developed a master-slave-type tele-micro-surgical system with an intelligent user interface [5]. An anti-shadow technique was used to present three-dimensional (3D) relative information on a two-dimensional (2D) display. Suturing of a 1-mm-diameter artificial micro-blood-vessel was demonstrated.

In tele-operation, the operator must depend on the feedback provided by sensory feedback systems such as video cameras to perform subsequent actions. His poor perception of the environment could result in a poor performance. For this point, researchers have been focusing

their research attention on another mode of DCS named as supervisory control. Supervisory control is based on a client-sever architecture shown in Fig. 1-4. In supervisory control, the user on a client station can give symbolic or analogical instructions remotely to a server computer attached to the manipulator instead of remotely guiding the tele-manipulator as he does in the tele-operation.

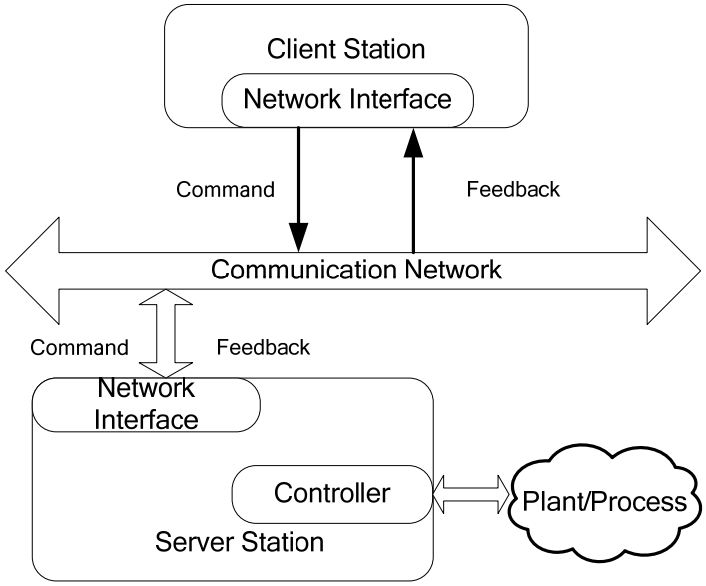


Fig. 1-4. Block diagram of a supervisory control system

Park and Sheridan [6] described a system developed for supervisory control of a tele-manipulator graphically simulated on an IRIS workstation. The system had two modes of operation: manual mode and supervisory mode. In the supervisory mode, the operator used a command menu and mouse to give symbolic instructions. The operator can specify intermediate locations that the manipulator tip is to pass through and select intermediate points that the hand tip should go through in the event of collision detection.

1.1.3 Networked Control Systems

With the development of computer and communication technologies in the last decades, sensors and actuators can now be equipped with network interfaces, being independent nodes in a real-time control system. This gives rise to an NCS with geographically distributed sensors, actuators, and controllers that communicate via a network [7–12]. The defining feature of an NCS is that information (about reference input, control input, plant output, etc.) is exchanged using a network among control system components (sensors, controller, actuators, etc.). Fig. 1-5 illustrates a typical block diagram of an NCS.

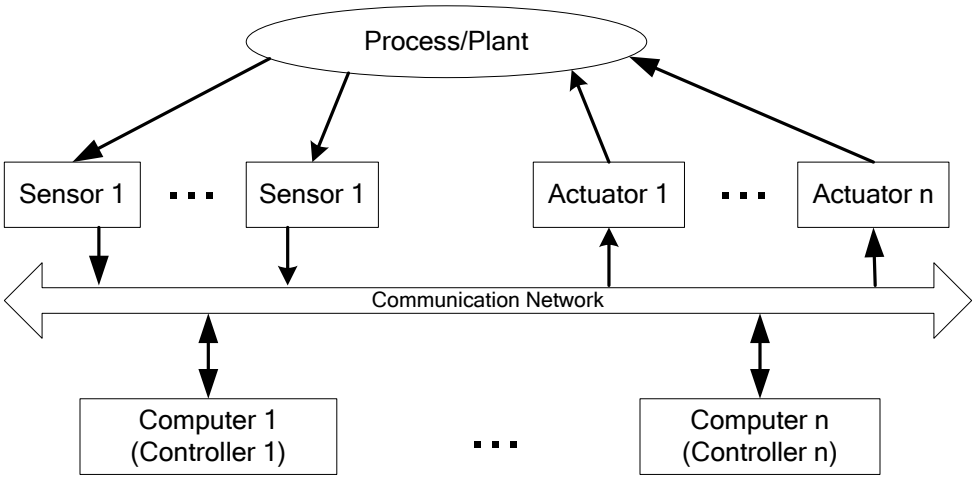


Fig. 1-5. Block diagram of an NCS

An NCS essentially comprises multiple nodes communicating with each other over communication networks. The primary advantages of an NCS are reduced wiring, easy of system diagnosis and maintenance, increased system agility, etc. An NCS offers the ability to locate the controllers for various processes at convenient locations that are easily accessible. For example,

in deep-sea exploration or a nuclear plant the controller can be placed in an easy-to-access office environment whereas the sensors and actuators can be placed at the required locations. In an event of malfunctioning of any control system, this control architecture also allows the operator to take corrective measures almost instantaneously. However, the insertion of a communication network in the feedback control loop makes the analysis and design of an NCS complicated. The success of this control scheme depends on the ability of the communication network in transferring data with stringent timing constraint and reliability requirements. In the development of NCSs, dealing network-induced time delays and data-packet losses in the communication networks has always been a key issue. An NCS with only one controller and one plant is denoted as a networked feedback control system as shown in Fig. 1-6 [8]. Networked feedback control systems via a wireless network were proposed in [9–10], there was no other network traffic in the dedicated wireless networks. The automation industry will soon reap the benefits of high-performance closed-loop control on distributed networks along with the development of reliable high-bandwidth networks and new protocols, such as Internet II and Rether (real-time Ethernet) [11–12]. Although this field is relatively new and still in its infancy, it has captured significant interests of many researchers worldwide [7].

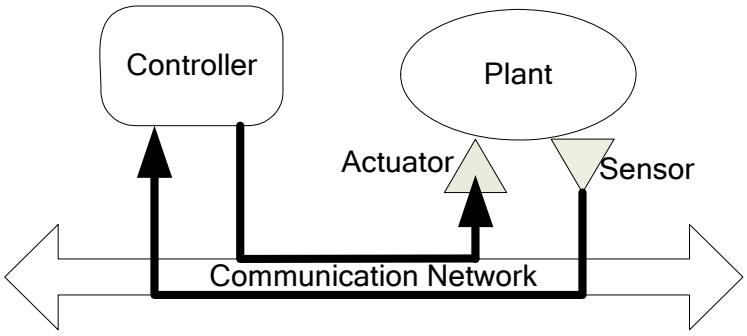


Fig. 1-6. Block diagram of a networked feedback control system

1.2 Internet-Based Control

The Internet as a communication medium provides cost-effective, flexible, and easy-to-access means for distributed control systems. If we can use the present form of the Internet as a communication medium for distributed real-time control, we can greatly enhance the advantages of this control architecture. Integration of the Internet will reduce the cost and the time involved in setting up the distributed real-time control system. Software applications can be used extensively to remotely monitor and control of the processes via the Internet. The incorporation of the Internet in the process control industry will also give engineers the freedom to distribute control tasks, sensors, and actuators to optimal locations. Thus, engineers will be able to monitor applications running in harsh environments that offer limited accessibility. Internet-based tele-operation and supervisory control have already been used in tele-robotics, remote manufacturing, tele-surgery, and distant education.

1.2.1 Internet-Based Tele-operation

Safaric *et al*. [13] developed a tele-operation-based method of education and training that involved the use of Virtual Environments for task planning. Instead of allowing the trainees to interact with the resources directly, this method requires them to configure the experiments using the simulated representation of a real-world apparatus. This configuration data can then be downloaded to the real work-cells. Hu *et al.* [14] discussed the use of cooperative Internet robots with interactive human-machine interface. Sato *et al.* [15] developed a master-slave-type-micro-teleoperation with intelligent user interface. Cybercut, developed at University of California at Berkeley provided a mechanical design and manufacturing service on the Internet [16]. Ho and

Zhang [17] argued the use of Java as an ideal implementation vehicle for internet-based tele-manipulation. A tele-operation system was built to control a three-fingered hand. The use of Java made the system-platform implementation independent and object-oriented.

1.2.2 Internet-Based Supervisory Control

Recently researchers started using the Internet to establish supervisory control for their tele-robots and test beds [18–20]. Mercury project [18] developed by Goldberg *et al*. was the first successful use of the Internet for supervisory control of an Internet-based robot. Their experiment allowed the general public to excavate objects in a sandbox using a telerobot. Luo *et al.* [19] used a supervisory control technique to develop a desktop rapid-prototyping system. Garcia *et al.* [20] developed a tele-robotic system using supervisory control based on a hybrid control approach. Srivastava [21] discussed the supervisory control via the Internet of a ball maglev system. In Internet-based supervisory control, the sensors, controllers, and actuators are located at the plant side as shown in Fig. 1-7. Like a tele-operation system, the control loop in supervisory control is also closed locally.

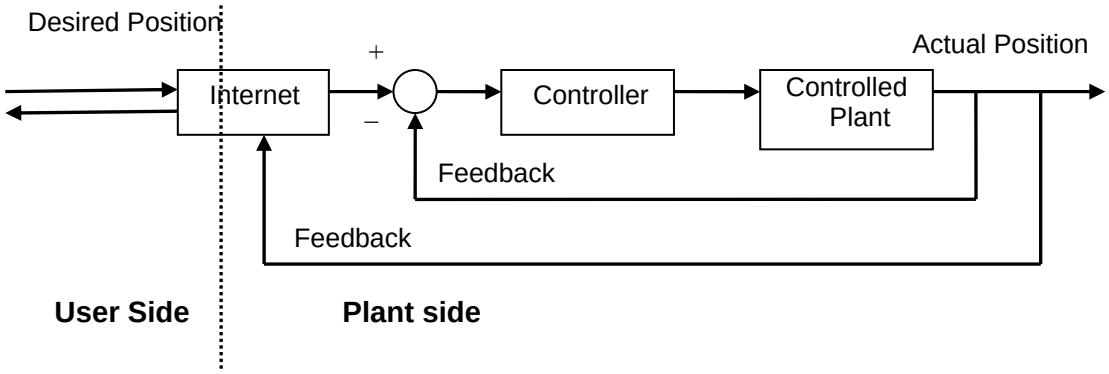


Fig. 1-7. Block diagram of a supervisory control via the Internet

1.2.3 Feedback Control over the Internet

A real-time feedback control system over the Internet consists of various sensors, actuators and controllers connected with each other via the Internet. Fig. 1-8 illustrates a block diagram of a feedback control over the Internet. The user side of the network runs the control algorithm. The controller receives data from various sensors and sends the data to the actuators via the Internet. With the Internet it is possible to set up all the sensors and actuators on the same communication network. This will eliminate the need to set up a dedicated communication network for different closed-loop control processes in the industry.

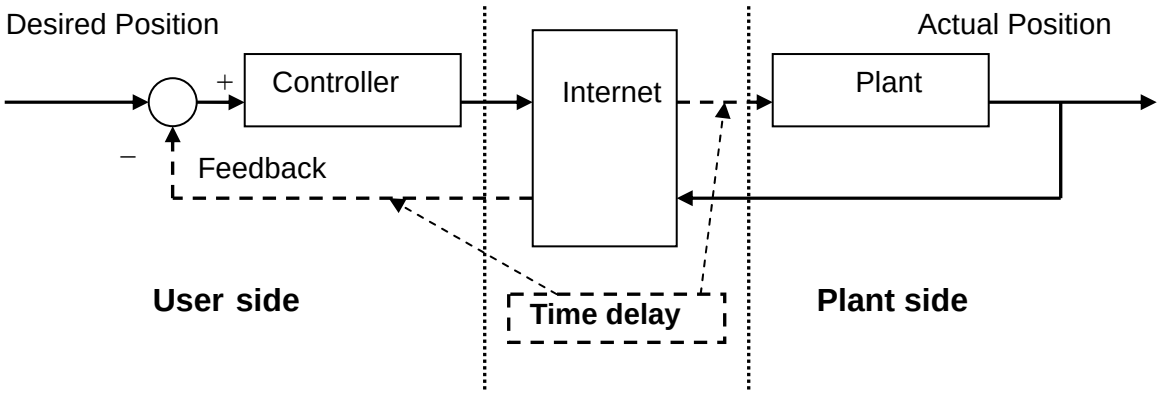


Fig. 1-8. Block diagram of a feedback control over the Internet

In feedback control over the network, the control loop is closed over the network. One application of feedback control over the network is a feedback control system with multiple clients as shown in Fig. 1-9. A sever controller and multiple clients share the network as a communication medium. This network-based control architecture provides good features such as easy installation, reconfiguration and can reduce the set-up and maintenance costs. However, how to do the network scheduling and bandwidth allocation for these clients to achieve optimal

quality of performance (QoP) is an open research issue and will be further investigated in this dissertation.

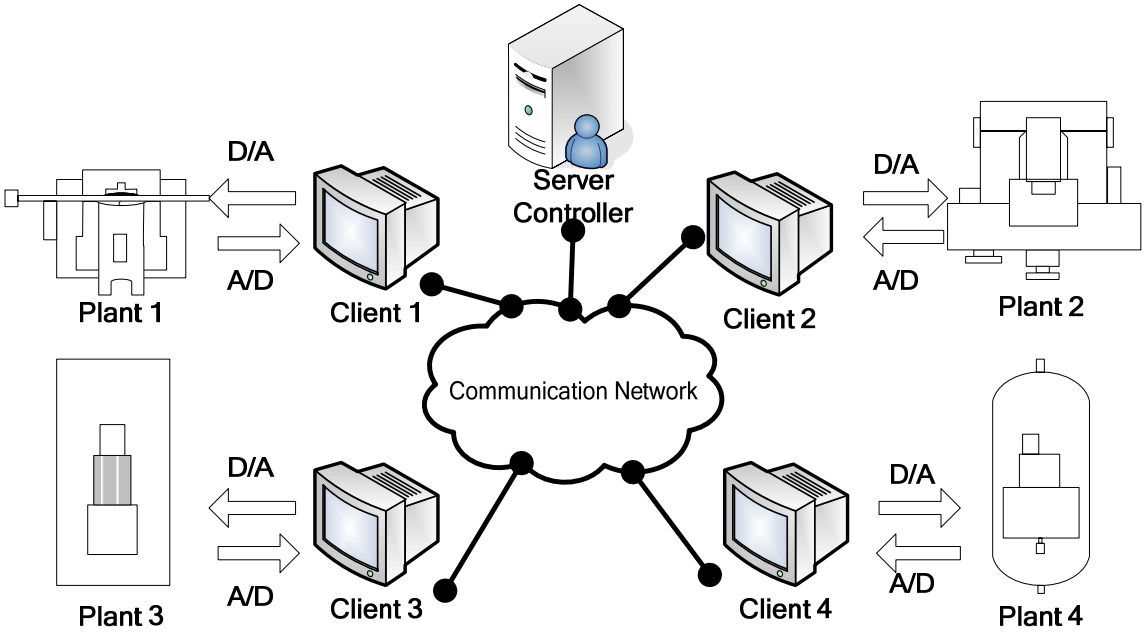


Fig. 1-9. Feedback control over network with multiple clients

1.3 Research Issues in Real-Time Control over Networks

The two control schemes via the Internet, namely supervisory control and real-time feedback control offer the ability to process data from various sensors and actuators in real time. Some of the key issues that need to be addressed for the satisfactory performance of a digital control system implemented via the Internet as a communication medium include:

1. The effect of the inevitable time delay associated with the Internet on the stability of the system. Network-induced time delays originate from the time sharing of communication media as well as additional functionality required for physical signal coding and processing.

Because of the nondeterministic factors during transmission process, Uniform sampling cannot be guaranteed. This results in a time-varying model, which makes system modeling and stability analyses difficult.

2. The unpredictability of the time delay associated with the Internet under different network protocols like transmission control protocol (TCP) and user datagram protocol (UDP). TCP is the most widely used transport protocol over the Internet. It is the protocol used for applications that require reliable communication. TCP is a confirmation-based protocol and introduces more time delay in data transmission. On the other hand, UDP is not a confirmation-based protocol. Thus, it reduces the overhead of retransmission time. Since UDP is not confirmation based, data might get lost during the transmission. The inevitable time delays and packet losses are detrimental to the real-time control system's performance.
3. Identification of fault when there is data loss in the system. During data transmission through the Internet, it is possible for the data to be lost. This data loss might be caused by defective sensor or the time delay that is far more than the sampling period of the system. When there is congestion in the communication network, some packets are dropped to either reduce the queue size or to inform the senders to reduce their transmission rates. Thus, it will be difficult to identify the exact reason for the data loss.
4. The functionality of real-time control over networks can be described as the sequence of four main operations (sampling, computation, data transmission, and actuation) that should be repeatedly executed, keeping strict timing, in order to deliver the expected performance. Not only the network could induce delays, but also the devices connected to the network could introduce latencies if they are not in a real-time operating environment. Therefore, to achieve the timing constraints and guarantee the control QoP, real-time control system design for the devices connected to the network must be developed.

5. In networked feedback control or NCSs, there are two classes of time delays and packet losses in both the sensor feedback and control feedforward paths. There is also the possibility of out-of-order data transmissions due to the nondeterministic factors during transmission process. How to develop compensation algorithms to deal with these problems and achieve control QoP is a very important issue.
6. Network scheduling and bandwidth allocation is an important issue in NCS design when a set of plants or processes are connected to the same network and share the limited bandwidth resource.

To address these important technical issues, two Internet-based control architectures, supervisory control and networked feedback control are developed and experimentally verified in this dissertation based on the works originally done by Srivastava [21] and Ambike [22].

1.3.1 Internet-Based Real-Time Control

For Internet-based tele-operation and supervisory control, dedicated interface software was used in earlier works [13–20]. To demonstrate the feasibility of Internet-based real-time control, a supervisory control via the Internet is implemented and verified in this dissertation based on the work of [21]. Srivastava [21] originally implemented a supervisory control of the ball maglev setup via the Internet by using a common gateway interface/hypertext markup language (CGI/HTML) interface. With this interface a client can connect to the server PC remotely via the Internet. The client can give the position commands, tune the controller running on the server PC, and receive the effect of the changes instantaneously.

For real-time control over the Internet, the control decisions and control functions usually have certain deadlines. If some of these deadlines are missed, the stability and performance of the real-time NCS could be negatively affected. Thus there is a need of real-time

operating systems (OS) for the control devices to reduce the time delay caused by device processing and to ensure these time-constrained events do not miss their deadlines. Commercially available OSs such as Windows, various versions of UNIX, and Linux are not real-time OSs. Ambike [22] originally implemented a real-time control system for the ball maglev setup. In this dissertation, to demonstrate the feasibility of Internet-based real-time control, a real-time solution of NCS design by using Linux with real-time application interface (RTAI) is implemented based on the work of [22]. Based on the real-time control system developed in [22], a simple Internet-based supervisory control of the ball maglev test bed by using a free secure shell client is also proposed in this dissertation.

1.3.2 Modeling and Stability Analysis of Networked Feedback Control

Conventional analyses of computer-controlled systems assume uniform sampling of the plant outputs, which means the samples are taken periodically with a constant time period. This assumption greatly simplifies the stability and performance analysis [23–24]. However, this assumption is not valid for the NCS analysis. Unlike conventional control systems, an NCS essentially comprises multiple nodes communicating with each other over communication networks. The network-induced time delays, data-packet losses, and out-of-order transmission have to be accounted for in modeling and analyzing an NCS. The issue of modeling, analysis, and design of an NCS with randomly varying delays is complicated and still an open research topic [7].

Åström and Wittenmark [24] provided a fundamental analysis of systems with delays in the feedback control loop. Both systems with delay less than one sampling period and systems with longer delays were considered. Only the case of constant delays was considered in their analysis.

Halevi and Ray [25] studied the integrated communication and control system (ICCS) with network-induced time-varying delay. They considered a continuous-time plant and discrete-time controller and analyzed the problem using the discrete-time approach and proposed a methodology named as the augmented deterministic discrete-time model methodology. The augmented system state vector consists of past values of the plant input and output, in addition to the current state vectors of the plant and controller.

Nilsson [26] addressed the problem of analysis and design of control systems when the communication delays vary in a random fashion. The two delay models for the communication network were developed. The first model consisted of random delays that were independent from transfer to transfer. In the second model an underlying Markov chain was used to generate the probability distributions of the time delays. In [26], the LQG-problem was solved by splitting it into a state feedback problem and an optimal state estimation problem.

In [27] the control of a continuous-time linear plant where the state sensor was connected to a linear controller/actuator via a network was addressed, and necessary and sufficient conditions for stability were derived for the presented setup in terms of the update time h and the parameters of the plant. However, only the delays in the feedback path between the sensor and the controller were considered.

Stability regions and the stability analysis of an NCS were proposed using a hybrid systems technique in [28–29]. Modeling of an NCS with multiple delays was considered in [30].

In this dissertation, the focus is on the modeling and analysis of NCSs where the effects of the network-induced time delay, data-packet loss, and out-of-order data transmission are all included. Deriving new delay-depended stability condition is also performed based on the new NCSs model.

1.3.3 Network-induced Time-Delay and Packet-Loss Compensation

Input delay is the most common form of network-induced time delays or latencies in an NCS. It is well known that the existence of time delays degrades the control performance and makes closed-loop stabilization more difficult because of the network-induced time delays within related sampling and actuation instants [28–29]. Studies have been done to ensure the stability of the system in the presence of time-varying delay in the communication network.

A technique to stabilize delay systems was proposed by Artstein [31] by transforming the delay system into a linear finite-dimensional system using an appropriate infinite-dimensional controller.

Liou *et al*. [32] proposed a time-driven sensor, a time-driven controller, and an event-driven actuator. The controller and the sensor had a time skew in their operation and the control value was calculated without the knowledge of new sensor signal. Due to an event-driven actuator the control signal was applied to the digital to analog (D/A) channel as soon as it arrived at the actuator node. These delayed signals were introduced in the discrete time augmented plant model. An augmented plant model was used to solve for linear quadratic (LQ) optimal controller. They also discussed the construction of state estimator for the case when all the states were not measured.

Ray *et al*. [33] extended the concept of a time-driven actuator and stochastic state estimator. The estimator was used to estimate the states that were unobservable. The state estimator is designed to minimize the variance of the state prediction error. The combination of a LQ controller and a minimum variance estimator was introduced as a delay compensated linear quadratic gaussian (DCLQG) controller.

Luck *et al*. [34] suggested a methodology to convert a time-varying system into a time-invariant system with the use of buffers. These buffers were introduced at the controller and

actuator nodes. The nodes and the buffers were all time-driven and synchronized in time. The buffers were used to store the data packets traveling through the communication channel. The buffers were made longer than the worst-case time delay present in the communication network. This allowed the data packets to be available at the controller and the actuator after fixed amounts of time making the system time invariant even in the presence of time-varying delay.

Chan and Ozguner [35] studied the Ford SCP Multiplex Network hardware. A queue was placed at the sensor node, and registers were appended at the controller node. The data packets sent from the sensor were appended with a time stamp and the queue size at that instant. With this method the controller node could reduce the uncertainty about the order of the sensor signal that was being used to generate the control input.

Nilsson [26] proposed an optimal control methodology. This optimal stochastic control methodology treated the effect of random network delays in an NCS as an LQG problem and assumed that time delay τ was less than the sampling interval h.

In [36], a robust state-predictive control strategy was proposed for discrete-time multi-input-multi-output (MIMO) systems with non-equal delays on signal buses. The input and output delays were taken into account in the control-law synthesis and the steady-state Kalman predictor design.

Zhang *et al*. [28] assumed that the full states were transmitted through the network and that they might be lost because of the dropped packets in the network. On the other hand, the control command was sent directly to the plant. With these assumptions, the authors used the stability analysis for asynchronous dynamical systems to find the maximum packet-dropping rate under which the overall system is stable.

In [37] the stability of a linear NCS in the presence of dropped packets was studied. Similar to [28], the controller was directly connected to the plant, so there was a direct link

between the controller and the plant. The stability analysis in [37] was based on the stability of Markov-jump linear systems.

Almost all the above research focused on dealing with either one-way time delays and data-packet losses or two-way time delays alone or data-packet losses alone. Network-induced time delays and packet losses may occur simultaneously in both the feedback path and the feedforward path, thus a more comprehensive compensation algorithm is needed that can deal with the time delays and data-packet losses in a unified way. The study in this dissertation is based on the work in [26] and [27]. The model in [27] is extended to deal with a framework of NCSs allowing independent round-trip communication delays and data-packet losses. The stochastic optimal estimator in [26] is extended to be a multi-step-ahead state estimator to compensate for the time delay longer than one sampling period and successive data-packet losses in both the control forward and the sensor feedback paths simultaneously.

1.3.4 Advanced Control Design of Feedback Control over Networks

Because the key for NCSs is that almost no local control action can be taken in isolation from the rest of the system and that design parameters of feedback control and real-time communication systems are interdependent, the successful design and implementation of an NCS requires an appropriate integration of control systems, real-time systems, and network communication systems through co-design. As the sampling period approaches zero, the performance of the digital system approaches that of a continuous-time system. To improve the QoC of NCSs, advanced control designs are presented in this dissertation with an emphasis on optimal co-design and robust control design for NCSs.

For an NCS, performance is a function of not only the sampling period, but also the traffic load on the network [38]. A system design chart was proposed in [38] as a guideline for

NCS design. Based on the design chart modified after [38], a quantitative method about how to determine the location of the performance degradation points in the NCS performance design chart is presented in this dissertation. The optimal working range of the sampling frequency can be determined based on the locations of these points.

For the H_∞ control of the input delayed system, a functional analytic technique was used by Foias *et al*. [39]. The standard H_∞ control problem for linear systems with delay was solved based on two Riccati equations by Keulen [40]. Kojima *et al*. [41] developed a robust controller with respect to uncertain input delay based on a Riccati-equation approach. Boyd *et al*. [42] emphasized that many problems arising in system theory could be cast into the form of linear matrix inequalities (LMIs). Niculescu [43] introduced an approach based on LMIs to derive sufficient conditions for the stabilization of systems with uncertain input delay. This dissertation focuses on robust control of NCSs with parameter uncertainty. Robust H_∞ control problems for NCSs with network-induced time delays and subject to norm-bounded parameter uncertainties are addressed in this dissertation.

1.3.5 Network Scheduling and Adaptive Control Co-Design

Network scheduling is also an issue in NCS design when many plants are connected to the same network as shown in Fig. 1-9. The key advantage of this NCS is that it provides the flexibility to quickly reconfigure its system architecture and to easily share information with other subsystems. Network traffic introduces inevitable delays. An effective way to reduce the negative effect of delays on the performance of NCSs is to reduce network traffic. Using deadbands to reduce communication in NCSs was proposed as a solution for network traffic reduction in [44]. Yook *et al*. [45] formulated a method that offers network traffic reduction in exchange for added computational cost of using estimators to predict the states of other systems

on the network. In [46], a traffic-smoothing method was proposed to reduce the packet-collision ratio. At each node, a traffic-smoother is installed between the transport layer and the Ethernet medium-access-control (MAC) layer and regulates its packet stream using a certain traffic-generation rate.

Traditionally, research on bandwidth allocation and scheduling techniques for NCSs focused on static strategies that ensure average control performance at the expense of permanently occupying the available bandwidth. Hong [47] and Hong and Kim [48] developed a scheduling algorithm to determine the sampling periods of multiple control loops with cyclic service discipline so that the performance requirement of each control loop was satisfied and the utilization of network resources is increased. Although very simple and effective, the scheduling method in [48] is not always economical and optimal because the exponential scalar configuration of the sampling period uses up much of the bandwidth resource. The work of extending the concept of the maximum allowable bound in [47] to the multidimensional cases was proposed in [49]. Branicky *et al*. [50] formulated a static optimal scheduling problem under both rate-monotonic-schedulability constrains and NCS-stability constraints. From the control perspective, the static bandwidth allocation method is an "open-loop" solution because once established at system set-up, the static scheduling will not be adjusted at run-time. Given sufficient bandwidth, the static bandwidth allocation can successfully guarantee real-time communication and meet the control requirements. However, due to network bandwidth limitation in some cases, not all control loops can simultaneously gain enough bandwidth allocation to provide the best possible control performance. Thus scheduling the network with an "open-loop" algorithm may cause critical messages to fail in timely transmission, degrading control performance or even leading to instability of certain control loops. Moreover, static techniques may not be efficient when changing conditions occur at the control-application or

network levels, because pre-assigned bandwidth resources could be underutilized and these underutilized resources could be made available to other applications.

To address these problems, this dissertation follows a “closed-loop” solution and focuses on developing a dynamic optimal-network-bandwidth-allocation (ONBA) algorithm that makes scheduling decisions based on the performance information feedback of each control loop. Following the methodology of feedback scheduling [51], the control requirements are integrated into the scheduling of the shared network.

Dynamic strategies for network scheduling in NCSs can be found in the literature. A control loop scheduling technique called large error first (LEF) was presented in [52]. The LEF algorithm uses feedback information from the application to assign communication bandwidth to each individual component. However, the relative importance of different control loops that may contribute to the whole system at different levels is not considered, and the implementation issues of LEF remain to be addressed. In [53] a dynamic arbitration technique called maximum-error-first with try-once-discard (MEF-TOD) was presented to grant network access to the control loop with highest error. However, the feedback methodology is not explicitly employed in [53]. Furthermore, both LEF and MEF-TOD techniques are referred to communication protocols and applying them in existing applications may be very expensive due to the requirements of excessive time and cost for system update. A method for optimal off-line scheduling of a limited communication resource used for control purpose was presented in [54].

Park *et al* [49] presented a scheduling method for NCSs to adjust the sampling period as small as possible, allocate the network bandwidth for three types of data, and exchange the transmission orders of data for sensors and actuators. The sampling adjustment was considered for the control analysis but was not performed according to the system dynamics and performances. In [55], bandwidth management of each control loop was done locally at run-time

according to the states of each controlled process, and control laws were designed to account for the variations on the assigned bandwidth. However, within this approach, the sampling periods are time varying, the quickly varying states may introduce abrupt and too frequent changes between sampling periods, which would imply excessive switching between different closed-loop modes (chattering). Further work [56] extended this dynamic bandwidth management to an optimal bandwidth allocation policy whose complexity may increase the requirement of computational power. The approach given in this dissertation is similar but significantly reduces the computational requirement.

The change of the system configuration might also change the time-delay signature of a networked device, thus change the network quality of service (QoS). Optimized QoC can be achieved if the networked controller can adaptively modify its control algorithm according to the QoS changes. This can be formulated as an adaptive control problem which can adjust its parameters on-line according to the changing network QoS. Bandwidth allocation problem and adaptive control problem are two different problems but follow similar design mechanisms. Bandwidth allocation problem searches optimal sampling period for control design based on control system QoP, while adaptive control problem searches optimal control parameters for control design based on network QoS. Thus they can be formulated as a co-design issue that is proposed in this dissertation. The objective of co-design is to design a networked controller that can adaptively modify the control algorithm according to the control QoP and network QoS.

1.4 Control Network Analysis

The successful design and implementation of an NCS requires an appropriate integration of co-design of the real-time control system and the network communication system. Choosing

appropriate control network is essential to the NCS design. It is necessary to understand the protocol message prioritization as well as the delay characteristics of the control networks in designing an NCS. The timing parameters, which will ultimately influence control applications, are affected by the network data rate, data or message size, and communication protocol. Generally, the key requirement for a control network is that a message should be transmitted successfully within a bounded time delay [57]. Compared with data networks, control networks have some distinguishing characteristics:

1. Most of the communications between controllers and sensors/actuators has a fixed sampling period and data are transmitted continuously, and thus the transmission rate is high.
2. Data size of each message is relatively small.
3. Since time delay has a serious effect on system performance, the real-time requirement of control networks is much more critical than that of data networks.

Control networks for feedback control purpose are based on the following protocols: Ethernet (IEEE 802.3), Token Bus (IEEE 802.4), Token Ring (IEEE 802.5), CAN (ISO 11898), and Wireless (IEEE 802.11). The characteristics of these prominent control network types are summarized in this section to explain how they influence the NCS performance. A more detailed comparison among Ethernet, Token-based, and CAN protocols can be found in [57].

1.4.1 DeviceNet (CANbus-Based)

In the CAN protocol, data are transmitted with message frames shown in Fig. 1-10 [58], and a message may be transmitted periodically, sporadically, or on demand [58]. Each message is given a priority that determines network access, and collisions do not destroy messages since the message with higher priority is delivered. Each message used in the CAN has a unique identifier defining the type of data and the identifier also automatically implies the message priority for

bus access [58]. However, the identifier does not indicate the destination or source address information in a message. If a node wants to transmit a message, it waits until the bus is free and then broadcasts the identifier of its message. Each node has an acceptance filter to decide whether to receive that message or not. A message is accepted only if an identifier of the message object is matched with the incoming message identifier [58]. Thus, a bound on the time delay for higher priority messages can be defined and used in analysis. CAN is not suitable for transmitting messages of large size although large messages can be transmitted using fragmentation [57]. DeviceNet is based on the CANbus protocol but does not use the same physical-layer interface as ISO 11898. It is developed originally for the automotive industry, and each message has a unique identifier defining the type of data such as engine speed, temperature, pressure or any other data. DeviceNet has a slow data rate of only 500 kbps with up to 64 devices on the bus [57].

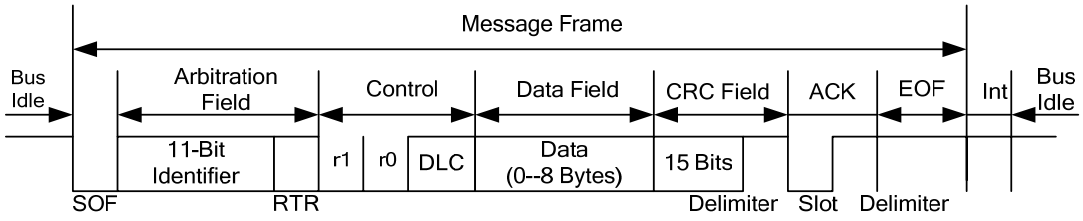


Fig. 1-10. Message frame of CAN [58]

1.4.2 ControlNet (Token-Passing-Bus-Based)

ControlNet is a deterministic network protocol used for time/mission critical applications. The message frame of ControlNet is shown in Fig. 1-11 [57]. In ControlNet, all nodes are arranged on a ring. A token is passed around the ring, and the node that holds the token is allowed to transmit data. This transmission continues until it is finished or a time limit is

reached, then the token is regenerated and passed on to the next node. Thus the maximum waiting time in a node before sending a message is the token rotation time. Message collisions never occur since only one node can transmit message at one time. In general, ControlNet is very efficient at high network loads and its deterministic nature defines a maximum bound on network-induced delays which makes time delay analysis easier. However at low network loads, a significant amount of time is spent passing the token around the logical ring [57]. In the case of an emergency, a node cannot gain access to transmit message until the token finishes its rotation around the logical ring. Under light to moderate network loads, ControlNet provides good and fair performance during increased loads. A refined token-based algorithm assures fair access with deterministic waiting time delays was proposed in [59].

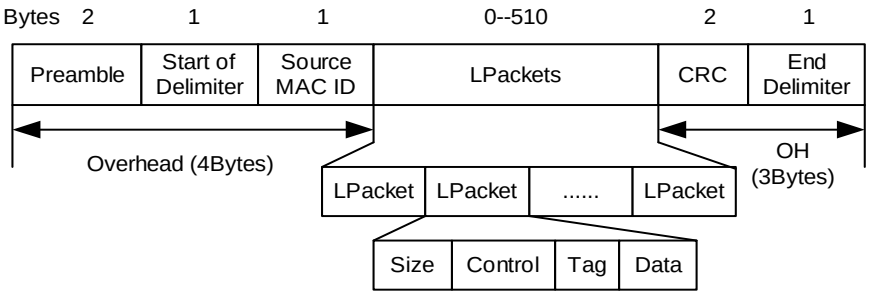


Fig. 1-11. Message frame of ControlNet [57]

1.4.3 Ethernet (CSMA/CD)

Ethernet does not support message prioritization and is not a deterministic protocol. Ethernet uses the carrier-sense multiple access with collision detection (CSMA/CD) mechanism to resolve the problem of contention in case of simultaneous data transmission [57]. If two nodes transmit data packets simultaneously, the packets collide. If a collision is detected, the two transmitting nodes wait a random length of time to retry transmission. The random length of time

is determined by the standard binary-exponential-backoff (BEB) algorithm. If 16 collisions are detected, the node stops transmitting, and then data-packet losses occur [60]. Thus in modeling, both time delays and packet loss need to be considered. Since Ethernet has low medium overhead and uses a simple algorithm for network operation, delays are small at low network loads [57]. Unlike ControlNet or DeviceNet, communication bandwidth is not wasted in message arbitration and message collisions lead to message loss in Ethernet [57]. The large frame size also makes Ethernet better suited to transmit large-size data with low frequency. The message frame of Ethernet is shown in Fig. 1-12. The speed standards of Ethernet include 10Mb/s, 100Mb/s, and 1Gb/s [60].

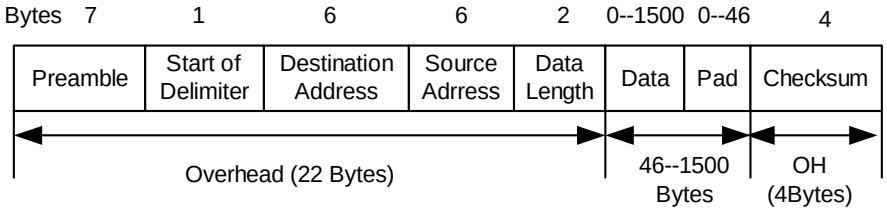


Fig. 1-12. Message frame of Ethernet [60]

Ethernet's contention-based mechanism makes it impossible to predict the network-induced delay. Switched Ethernet can provide deterministic delays by eliminating message collisions, but its high price has restricted its implementation in industry [61]. Several software changes have been suggested so that the network-induced delay could be bounded. Kweon *et al*. [61] developed a traffic-smoothing method to decrease the packet-collision ratio on the network which requires minimal changes in the OS kernel. The traffic smoother regulates the node's packet stream using a certain traffic-generation rate to eliminate collisions effectively. Venkatramani [12] proposed the approach of a timed-token bus to provide bandwidth guarantees. They proposed a software-based protocol called RETHER (Real-time Ethernet).

1.4.4 Wireless Network

Wireless networks have also been investigated as control networks. The performance analysis of the wireless-medium-access-control (WMAC) protocol and the remote-frame-medium-access-control (RFMAC) protocol for a wireless controller area network (WCAN) was presented in [62]. In [63], the wireless Ethernet (802.11) standard was modified and used to prioritize the carrier sense multi-access with collision avoidance (CSMA/CA), and thus message collisions were reduced. Later work [64] applied the wireless 802.11 control and scheduling algorithm to the control of a physical plant. Ploplys *et al*. [10] developed a distributed-control system for a Furuta pendulum over a wireless communication network based on 802.11b.

Wireless network used in networked feedback control is a dedicated network, and there is no other traffic and functionalities sharing the network resource. Thus the network-induced delay problem is not as severe as those in networked feedback control systems using other networks as communication media.

1.4.5 Summary of Static Parameters of Control Networks

Typical static parameters of the three control networks discussed above are summarized in Table 1-1[57]. Ethernet is potentially the most practical control network solution because of its low cost, availability, and higher communication rates, although it is not designed to transmit short messages with real-time requirements. In terms of implementation, Ethernet is not yet ready for manufacturing automation because its hardware was not designed to withstand stress, vibrations, or noise [65] and it has unpredictable delay and packet loss characteristics. However, under low traffic loads, Ethernet delivers fast data transmissions with almost no latency [57]. As a result, an Ethernet network structure may be suitable for control when the network is relatively

uncrowded. An example using Ethernet in networked control is given in [66], where UDP over a switched Ethernet was shown to exhibit good performance characteristics that were sufficient for substation automation. Based on the networked-feedback-control framework shown in Fig. 1-6, a real-time closed-loop control system via the Ethernet is designed and implemented and experimental results are presented in the following chapter.

Table 1-1[57] Typical parameters of control networks

Parameters	ControlNet	DeviceNet	Ethernet
Data Rate (Mbps)	5	0.5	10
Maximum length (m)	1000	100	2500
Maximum Data Size (byte)	504	8	1500
Minimum Data Size (byte)	7	47/8	73

1.5 Dissertation Contributions

One of the main objectives of this dissertation is to present a framework for the modeling, design, stability analysis, control, and bandwidth allocation of real-time control over networks. Two co-design algorithms for real-time NCSs are proposed. Co-design means that NCS design should consider the control network's characteristics and real-time control specifications at the same time. The optimal co-design algorithm utilizes both network and control parameters to reveal the existence and location of the performance degradation points which limit the operating range of the sampling frequencies. The network scheduling and adaptive control co-design algorithm is proposed to design a networked controller that can adaptively modify the control algorithm according to the control QoP and network QoS. In addition, a ball maglev system [67] is used as an NCS test bed for the experimental verification.

1.5.1 Experimental Verification of Internet-Based Real-Time Control

Two Internet-based control architectures, supervisory control and networked feedback control are developed and experimentally verified in this dissertation based on the works originally done by Srivastava [21] and Ambike [22]. A new Internet-based supervisory control based on the real-time system developed in [22] and a secure shell client is proposed and implemented. Thus one of the main contributions of this dissertation is the experimental verification of the feasibility of Internet-based real-time control.

1.5.2 New System Model and Stability Analysis for Networked Feedback Control

The study on the modeling and analysis of NCSs in this dissertation is based on the consideration of the nondeterministic factors during data transmission. How to build a general form of the NCS model where the effects of the network-induced time delay, data-packet loss, and out-of-order data transmission are all included is presented in this dissertation. A new delay dependent stability criterion and the upper bound of time delay are derived through a Lyapunov functional approach. The appropriate co-design integration of control systems, real-time systems, and network communication systems proposed in this dissertation is based on this model.

1.5.3 Compensation Algorithms for Time Delay and Packet Loss

The study of delay/data-loss compensation in this dissertation is based on the work in [26] and [27]. The model in [27] is extended to deal with a framework of NCSs allowing independent round-trip communication delays and data-packet losses. The stochastic optimal estimator in [26] is extended to be a multi-step-ahead state estimator to compensate for the time delay longer than one sampling period and successive data-packet losses. Two compensation

algorithms for the time-delay and packet-loss in both the control feedforward and the sensor feedback paths simultaneously are proposed and verified in this dissertation.

1.5.4 Robust Control Design and Optimal Co-Design for NCSs

Robust H_∞ control problems for NCSs with network-induced time delays and subject to norm-bounded parameter uncertainties are addressed in this dissertation. There are not many research results on NCSs considering parameter uncertainty and time delay simultaneously. Similar approaches were presented by Lee and Lee [68] and Li *et al*. [69]. However, they focused on continuous-time models and constant time delays, which is not the case in NCSs.

Based on the design chart modified after [38], this dissertation presents a quantitative method about how to determine the location of the performance degradation points in the NCS performance design chart. The optimal working range of the sampling frequency can be determined based on the locations of these points. The optimal working range of the sampling frequency proposed in this dissertation can be used as a guideline for NCS design, which is verified by the design of the ball maglev NCS test bed. How optimal controllers can be designed for QoC optimization is also described in this dissertation.

The robust control design and optimal co-design methodologies proposed in this dissertation are experimentally verified with the real-time NCS test bed implemented in [22].

1.5.5 Optimal Network Bandwidth Allocation Algorithm for NCSs

This dissertation presents a dynamic ONBA algorithm for NCS design that optimizes overall control performance and reduces network-bandwidth usage. Network-bandwidth is dynamically assigned to each control loop according to the control performance information of

each control loop. The first advantage of this algorithm is that the computational power requirement is low and the allocation of bandwidth to control loops can be done locally at run time according to how far the control loops are from their equilibrium. Thus it can be used to enable existing NCSs to provide satisfactory QoP under resource constraints. The second advantage is that this algorithm does not cause excessive switching between different closed-loop modes (chattering) which may lead to instability. Simulation results are presented to show that this algorithm improves control performance and uses less bandwidth with respect to the static strategy.

1.6 Dissertation Overview and Related Publications

This dissertation contains seven chapters. Chapter I describes the various modes of computer controlled systems and presents an introduction of a relatively new area of real-time control via the Internet. The research issues about real-time control over networks are discussed and the contributions of this dissertation are presented.

Chapter II presents the implementation and experimental verification of Internet-based real-time control of a ball maglev test bed. Two control architectures are experimentally verified. The hardware setup and software structure of an Internet-based supervisory control based on the work of [21] are explained. This chapter also introduces the development of a novel real-time operating environment enabling closed-loop real-time control over the Ethernet based on the work of [22]. An Internet-based supervisory control of a maglev test bed based on the real-time system and a secure shell client is presented.

Chapter III presents the dynamic modeling of real-time control over networks. In this model, the effects of the network-induced time delay, data-packet loss, and out-of-order data

transmission are dealt with simultaneously. A new sufficient stability condition based on a Lyapunov method is also derived in this chapter.

Chapter IV presents two delay/data-loss compensation approaches to overcome the adverse influences of stochastic time delays and packet losses encountered in real-time control over networks. The first approach is based on model-estimation. The second approach is based on predictors and timeout scheme.

Chapter V discusses advanced control designs for NCSs. Advanced control theories such as robust control and optimal control are investigated and applied to improve the QoC of the NCSs. A modified system design chart of NCSs based on the work in [38] is proposed in this chapter and can be a useful guideline for choosing the optimal network and control parameters in designing an NCS.

Chapter VI formulates an optimal network scheduling and adaptive control co-design problem for NCS design. An ONBA algorithm is presented to optimize overall control performance and reduce network bandwidth usage.

Chapter VII concludes this dissertation summarizing its achievements and provides suggestions for possible future work.

Much of the research in this dissertation has previously been published in the papers and presentations listed here in chronological order:

- K. Ji, W.-J. Kim, and A. Ambike, "Control strategies for distributed real-time control with time delays and packets losses," in *Proc. of ASME International Mechanical Engineering Congress and Exposition*, Paper No. 61733, November 2004 (appears in Chapter IV).
- W.-J. Kim, K. Ji, and A. Ambike, "Networked real-time control strategies dealing with stochastic time delays and packet losses," in *Proc. of 2005 American Control Conference*, pp. 621–626, June 2005 (appears in Chapter IV).

- A. Ambike, W. -J. Kim, and K. Ji, "Real-time operating environment for networked control systems," in *Proc. of 2005 American Control Conference*, pp. 2353–2358, June 2005 (appears in Chapter II).
- K. Ji and W.-J. Kim, "Robust control for networked control systems with admissible parameter uncertainties," in *Proc. of ASME International Mechanical Engineering Congress and Exposition*, Paper No. 81551. Nov. 2005 (appears in Chapter V).
- K. Ji and W.-J. Kim, "Real-time control of networked control systems via Ethernet," *International Journal of Control, Automation, and Systems*, vol. 3, no. 4, pp. 591–600, Dec. 2005 (appears in Chapters I and II).

CHAPTER II

INTERNET-BASED REAL-TIME CONTROL*

This chapter describes the implementation of Internet-based real-time control of a ball maglev test bed. The implementations of two control architectures are based on the works in [22] and [21]. Two control architectures as supervisory control and networked feedback control are implemented and experiment results are presented to verify the feasibility of Internet-based real-time control.

2.1 Introduction

Supervisory control is a technique of remotely monitoring and enabling a computer to connect to an experiment and controlling it from a distance. The supervisory control via a communication network is based on the client/server architecture. It can be classified as a DCS. In this mode of control only the user-defined commands are sent via the communication network. Supervisory control has been widely applied in the areas of tele-robotics, under-sea environments hazardous environments and space programs. In these situations the controller is located along with the process to be controlled and the feedback loop is closed locally. Engineers can monitor these processes remotely and get feedback from the process in real time via the communication network. In an event of emergency where the controller is not performing efficiently or satisfactorily the operators can take corrective measures. These corrective commands are transmitted via the communication network to the controller and are implemented

in the next sampling cycle of the controller. The response of the system due to these changes is sent through the communication channel from the process to the operator in the same sampling cycle. The hardware setup and software structure of the supervisory control proposed in [21] are presented in this chapter to verify the feasibility of Internet-based supervisory control.

In networked feedback control, decision and control functions can be distributed on the network. That is, the control loop is closed over the network. Thus, a new constraint must be accommodated in the design of a real-time control over networks—the limited bandwidth of the communication network. Time delays caused by network data transmission and device processing are inevitable. To reduce the time delay caused by device processing, these control functions should have certain deadlines. If some of these deadlines are missed, the stability and performance of the control system could be negatively affected. Thus there is a need of real-time operating systems for the devices to ensure these time-constrained events do not miss their deadlines.

A real-time system can be defined as a system that responds to externally generated stimuli within finite and specified period of time [70]. An efficient real-time system produces correct results at proper time. The real-time systems can be classified into hard real-time systems and soft real-time systems. The system in which meeting all the time requirements is mandatory is known as a hard real-time system. In a soft real-time system, it is required that almost all the time requirements be met. But, the soft real-time system functions properly if a few deadlines are occasionally missed.

The Internet-based real-time networked feedback control architecture implemented and verified in this chapter is based on the real-time control environment proposed in [22]. A new simple Internet-based supervisory control based on this real-time environment is also presented in this chapter.

2.2 Supervisory Control via the Internet

With the widespread popularity of the Internet and its ease of communication real-time supervisory control of the processes can be implemented via the Internet. The supervisory control of a ball maglev test bed via the Internet proposed in [21] is introduced in this section.

2.2.1 Ball Magnetic-Levitation Test Bed

Stephen C. Paschall, II developed a ball maglev system shown in Fig. 2-1 as his senior honors thesis under the supervision of Won-jong Kim [67]. The objective of this maglev system is to levitate a steel ball at a predetermined steady-state equilibrium position with an electromagnet.

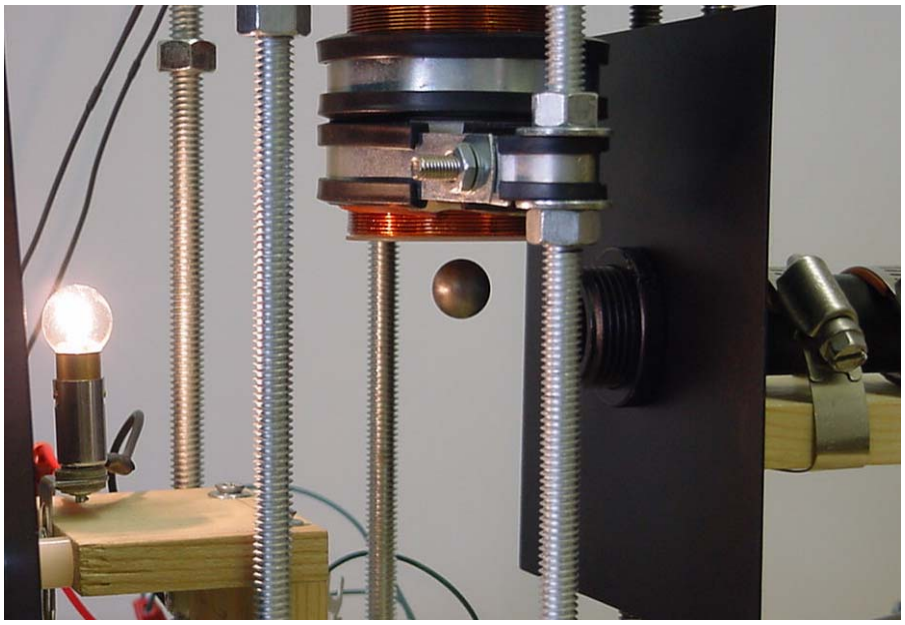


Fig. 2-1. Single-actuator magnetic ball levitation setup [67]

The framework of the supervisory control via the Internet proposed in [21] is shown in Fig. 2-2. The ball maglev setup is connected to the host Pentium IV personal computer (PC) that runs the Internet Information Services (IIS) 5.0 Web server on the Windows 2000 Professional OS. The maglev setup is controlled using a CGI/HTML interface with which a client can give the position commands remotely to move the steel ball within its travel range. The control parameters, such as the gain and the locations of poles and zeros can also be tuned on-line in real time. The client immediately receives the results from the changes in the control parameters or commands he/she has made.

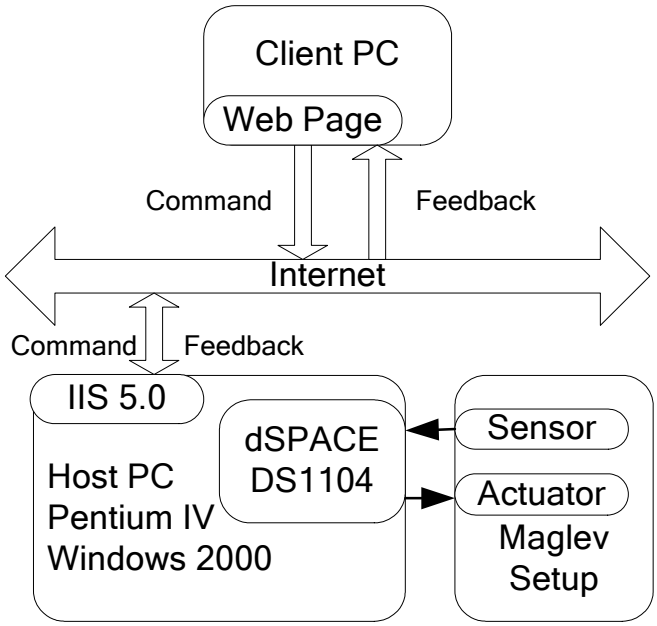


Fig. 2-2. Framework of the supervisory control of the test bed via the Internet [21]

2.2.2 Software Architecture

A real-time control algorithm was implemented in software on a dSPACE DS 1104 DSP controller board so that it can easily communicate with the CGI environment and obtain the

corresponding system responses in real time from the ball maglev test bed [71]. The client can access this Web page of the maglev system by typing the domain name of the host PC in the Web browser. An HTML page served by the host PC gets downloaded on the client PC so that the client can input his/her control parameters. Both the transactions of control parameters from the Internet and to the control algorithm are done simultaneously to save computation time. The software architecture for the supervisory control of the Maglev system via the Internet is depicted in the Fig. 2-3.

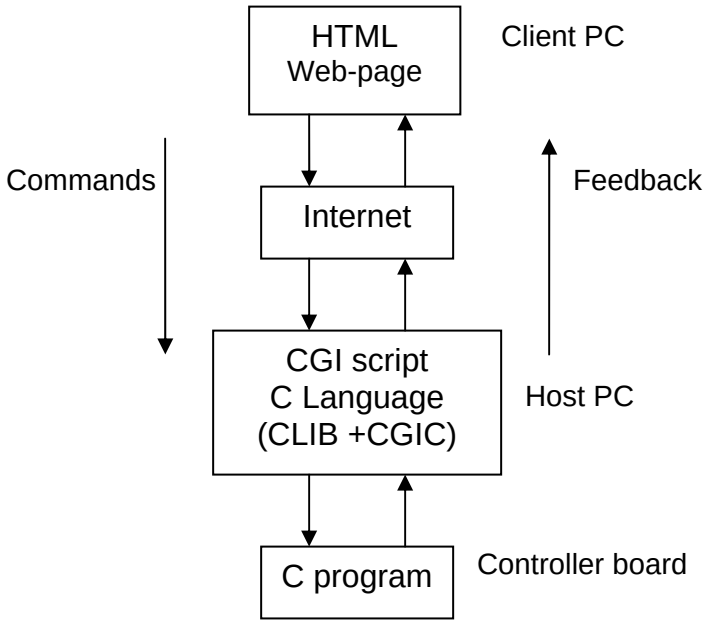


Fig. 2-3. Software architecture for the supervisory control via the Internet [21]

In supervisory control of the maglev system, stability of the system is not affected by the Internet. This is due to fact that no sensor data or control data traveled through a communication network and real time control is achieved by the local system with dSPACE controller board.

2.3 Real-Time Operating System for Networked Feedback Control

This section presents the development of a real-time networked feedback control system based on the work in [22]. The need of real-time operating environments and a discussion on various factors affecting the selection of real-time operating environments are elaborated. A real-time control system over the Ethernet based on client-server architecture and RTAI 24.1.12 with Redhat Linux 7.3 is implemented and experimentally verified.

2.3.1 Need and Selection of Real-Time OS

From the discussion of the previous chapter, we concluded that Ethernet is potentially the most practical network solution for an NCS design. Various communication protocols such as TCP/IP are used in Ethernet. However, the use of an Ethernet local area network (LAN) in NCSs poses several technical challenges including dealing with network latencies. Fig. 2-4 shows the time-delay components of the network latency in a periodic client-server communication process. Table 2-1 gives the nomenclature of the time delay components shown in Fig. 2-4. We assume that there is no data collision on the network.

Table 2-1. Nomenclature of time-delay components [22]

Symbol	Description
TC_{prep}	Time taken by the client to prepare the request message
TC_{wait}	Time spent by the client waiting for network access
$TC_{transmit}$	Transmission time from the client to the server
$TS_{process}$	Time taken by the server to process the request
TS_{prep}	Time taken by the server to prepare the reply message
TS_{wait}	Time spent by the server waiting for network access
$TS_{transmit}$	Transmission time from the server to the client
$TC_{process}$	Time taken by the client to process the reply
T	Total period of the process on the client side

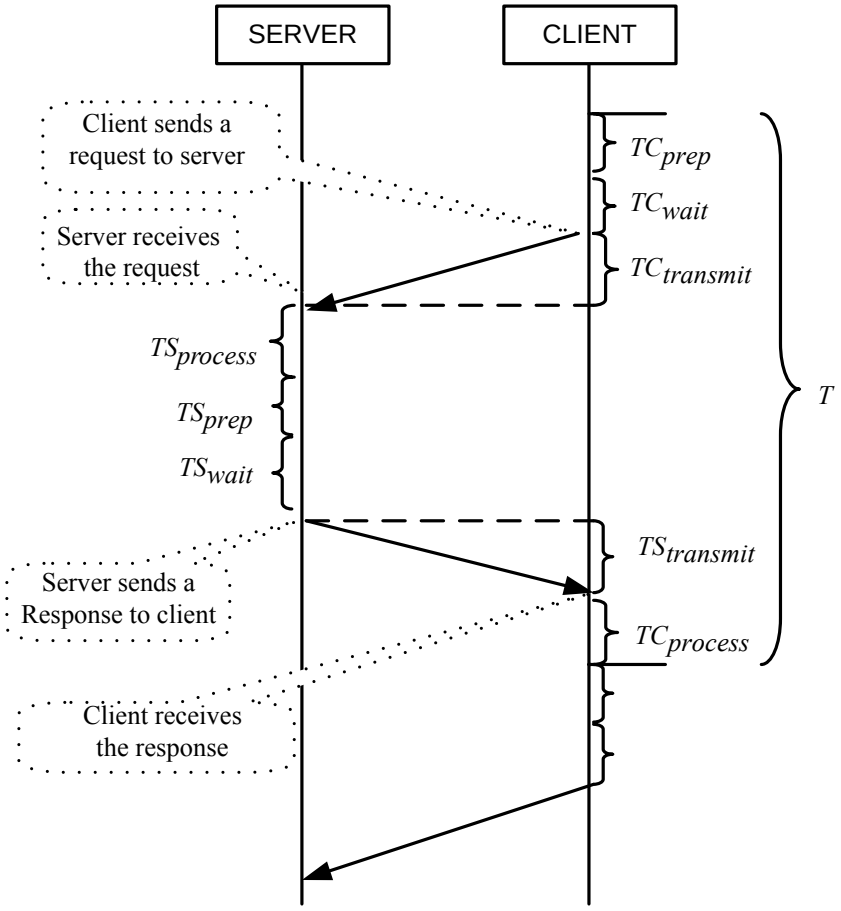


Fig. 2-4. Time-delay components of the network latency in a periodic client-server communication process [22]

As shown in Fig. 2-4, a typical client-server communication process in an NCS is periodic with a sampling period T. The total communication process is required to be completed in one period. The total communication time or the latency T_{total} is given by [22]

$$T_{total} = TC_{prep} + TC_{wait} + TC_{transmit} + TS_{process} + TS_{prep} + TS_{wait} + TS_{transmit} + TC_{process} . \quad (2.1)$$

In the current research, the ball maglev system shown in Fig. 2-1 is used again as the test bed to verify the feasibility of real-time control over the Ethernet. We can apply the client-server

architecture to a closed-loop NCS as shown in Fig. 2-5. The client side of the architecture includes the test bed. The server side implements the controller. The request message sent by the client to the server can carry the sensor data and the response message sent by the server to the host can carry the control data. The processing of the message on the server side is the calculation of control data using the sensor data. For this kind of closed-loop networked control system to remain stable, the events have certain deadlines. If these deadlines are missed, the stability of the control system is affected. In the current research, the ball maglev system is open-loop unstable. It has strict time requirements. It was found that for 333.333 Hz, the feedback loop should be completed in 1.42 ms [21]. If this deadline is missed, the system becomes unstable. Thus, the sampling should be done at a certain fixed frequency, e.g., 333.333 Hz. After every 3 ms, a fresh sample of the sensor data is taken. This sensor data is then transferred to the server to calculate the control in the form of a message. The creation of a message to be sent to the server is dependant on the sensor sampling. So, this process of sampling the sensor data is a real-time process. It is the responsibility of the OS environment to ensure that a sample is taken every 3 ms. In order to ensure that the time constrained events happen at correct times, a real-time operating environment is needed [22].

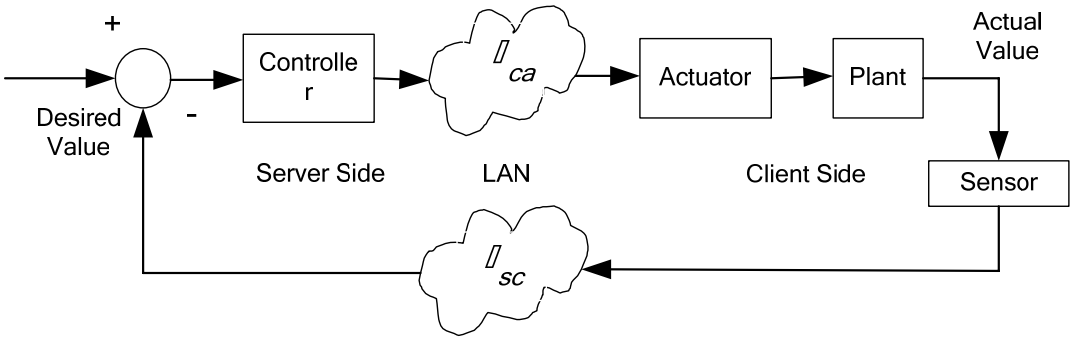


Fig. 2-5. Block diagram of a real-time NCS test bed based on a client-server architecture [72]

After the natures of available networks were studied, it was concluded that an appropriate real-time OS is required to reduce the time delay caused by device processing and to successfully implement a distributed architecture. Commercially available OSs such as Windows 2000, various versions of Unix and Linux are not real-time OSs. The Linux real-time application interface (RTAI) [73] was developed as a real-time operating environment solution at Dipartimento di Ingeneria Aerospaziale Politecnico di Milano (DIAPM). RTAI modifies the Linux kernel to make it a real-time operating environment. RTAI offers the same services as the Linux kernel core, adding the features of a real-time OS. Compared to the commercially available real-time OSs, RTAI's performance is very competitive [73]. Table 2-2 summarizes the typical performance of RTAI.

Table 2-2. RTAI's typical performance [73]

Context switch time	4 μs
Interrupt response	20 μs
Maximum periodic task rate	100 kHz
One-shot task rate	30 kHz

Ambike [22] proposed a real-time control system for the ball maglev setup by using Linux 7.3 with RTAI 24.1.12. Two timing tests were performed to observe the difference of the performances between RTAI and non-real-time OSs [74]. The following two paragraphs present the results of these two tests.

The smallest amount of time that can be precisely measured on an OS is known as its clock resolution. The time required to read a clock is typically much less than its resolution, and many consecutive clock access functions can be executed before the value returned by the clock changes. This principle was used in the first test. The number of times the clock was accessed

before the change in value returned by the clock access function was recorded and then plotted. Fig. 2-6 represents the results of the first test on Windows 2000, Redhat Linux 7.3, and Redhat Linux 7.3 with RTAI 24.1.12. Significant variations in Fig. 2-6(a) and (b) indicate some other OS activities that are not deterministic. In Fig. 2-6 (c), the straight line denotes that there was no significant non-deterministic OS activity in Linux with RTAI.

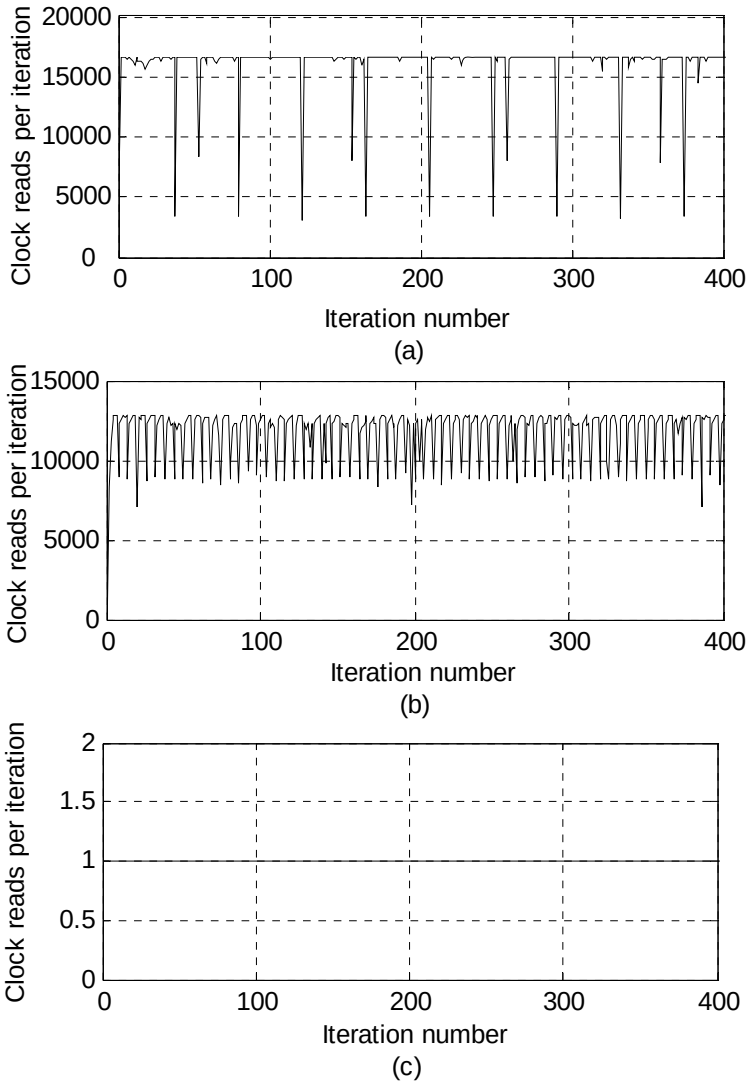


Fig. 2-6. Plots of the number of clock reads per iteration for the first timing test on (a) Windows 2000, (b) Redhat Linux 7.3, and (c) Redhat Linux 7.3 with RTAI 24.1.12

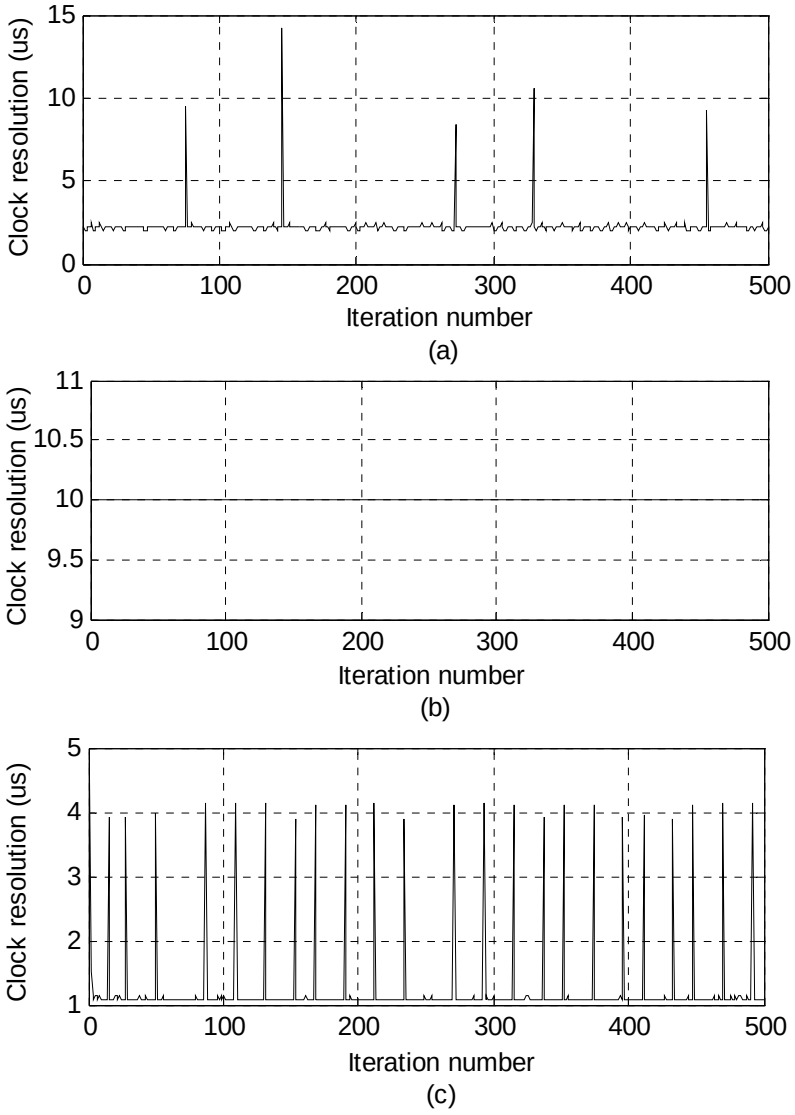


Fig. 2-7. Plots of the clock resolutions obtained for the second timing test on (a) Windows 2000, (b) Redhat Linux 7.3., and (c) Redhat Linux 7.3 with RTAI 24.1.12

In the second test, the clock resolution was calculated and plotted over several iterations. Fig. 2-7 presents the results of the second test on Windows 2000, Redhat Linux 7.3, and Redhat Linux 7.3 with RTAI 24.1.12, respectively. From Fig. 2-7, we can see that the clock resolution of RTAI is much better than that of Windows and Linux alone. Although the spikes in Fig 2-7 (c)

denote the variation in the clock resolution from 1 μs to 4.1 μs, the clock resolution reported is consistently less than 5 μs. These two simple tests demonstrated the non-real-time characteristics of the two popular OSs: Windows 2000 and Linux.

2.3.2 Design and Implementation of a Real-Time Networked Feedback Control

Linux control and measurement device interface (comedi) [75] is a free software project for tools, libraries, and drivers for various forms of data acquisition, and provides a collection of drivers for a variety of common data acquisition plug-in boards. It is used to provide the hardware-software interface [22]. It works with the standard Linux kernel as well as the real-time extensions such as RTLinux and RTAI. National Instruments' PCI-6025E is the data-acquisition board for the experiments [22].

Selection of network protocol for communication is an important part of NCS design. The two dominant choices are TCP and UDP. TCP was specifically designed to provide a reliable end-to-end byte streams over any unreliable network [60]. TCP provides various services like stream data transfer, reliability, efficient flow control, full duplex operation, multiplexing, etc. Handshaking signals are used for making and breaking the connection. Parameters such as sequence numbers are initialized to help ensure ordered delivery and robustness. If time delay is encountered, the data is retransmitted from the sender. Timers and acknowledgment messages are used to detect this time delay or data loss. Check sums are used to detect data corruptions that might occur during the data transfer. Although TCP is a very reliable protocol, it has some disadvantages. Due to various services such as error checking and ordered and reliable data delivery, it has large overheads leading to time delay in the communication. In the event of congestion, the data are lost more frequently, so more retransmissions are done by TCP, increasing the overheads. For closed-loop control over the network the added reliability provided

by TCP may not be worth the cost of the network delays it introduces.

UDP is an alternative provided by the TCP/IP protocol suite. Data transfer with UDP is not connection oriented. UDP does not provide additional services such as ensuring ordered data delivery and robustness as provided by TCP. It is therefore known as a best-effort network protocol. Although UDP is less reliable, it has fewer overheads and introduces less network delays. Ploplys *et al*. [10] concluded that the UDP, an unreliable but faster protocol, was better suited for real-time control over a dedicated wireless computer network. The general properties of TCP and UDP are compared in table 2-3.

Table 2-3. General properties of TCP and UDP

Properties	TCP	UDP
Data Flow Control	Yes	No
Message Boundaries	No	Yes
Connection Oriented	Yes	No
Positive Acknowledgement	Yes	No
Data Checksum	Yes	Optional
Duplicate Detection	Yes	No
Timeout and Retransmission	Yes	No

2.3.3 Experimental Verification

The mathematical model between the PWM output (V) and the position sensor output (Y) is described by a second-order transfer function [67].

$$G(s) = \frac{Y(s)}{V(s)} = \frac{-0.02792}{0.0086s^2} \tag{2.2}$$

Using this ball Maglev test bed and the Ethernet LAN in our lab, a real-time NCS shown

in Fig. 2-8 is constructed. The system configuration is the same as shown in Fig. 2-5. The plant PC with NI PCI-6025E as the data-acquisition card enables the ball maglev test bed to send out sensor data and receive control data through the LAN. The controller PC receives the sensor data, computes the control data, and then sends out the control data through the same LAN. Linux with RTAI is implemented on both PCs to ensure the time-constrained events like sampling and actuating do not miss their deadlines [22].

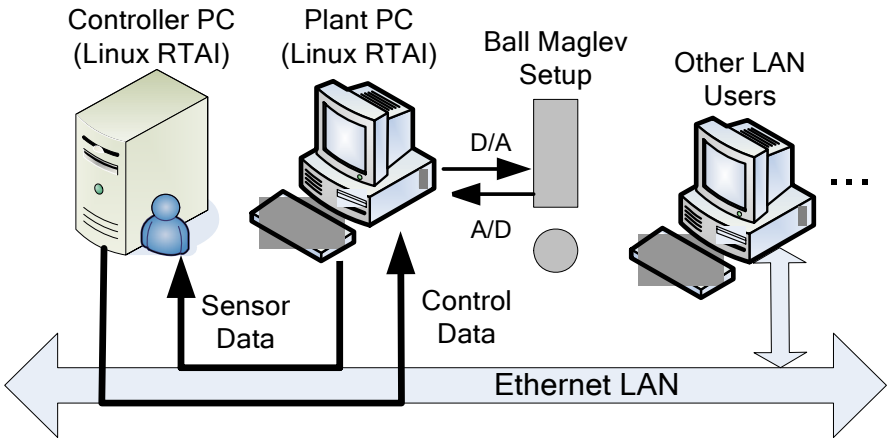


Fig. 2-8. Real-time NCS over an Etherent LAN

A digital lead-led controller designed in [22] to stabilize this set up is given as

$$D(z) = 4.15 \times 10^4 \left[\frac{z^2 - 1.754z + 0.769}{z^2 - 0.782z - 0.13} \right]. \tag{2.3}$$

The performance of our real-time operating environment was tested herein. First, the single-actuator ball maglev system was controlled using a PC with a 1.7-GHz Pentium IV processor and a data acquisition board PCI-6025E from National Instruments. Windows 2000

was used as the OS to interact with this hardware. Fig. 2-9 shows the system response of tracking a sinusoidal position command. The maximum command frequency that this control system could follow was 0.7 Hz as shown in Fig. 2-9.

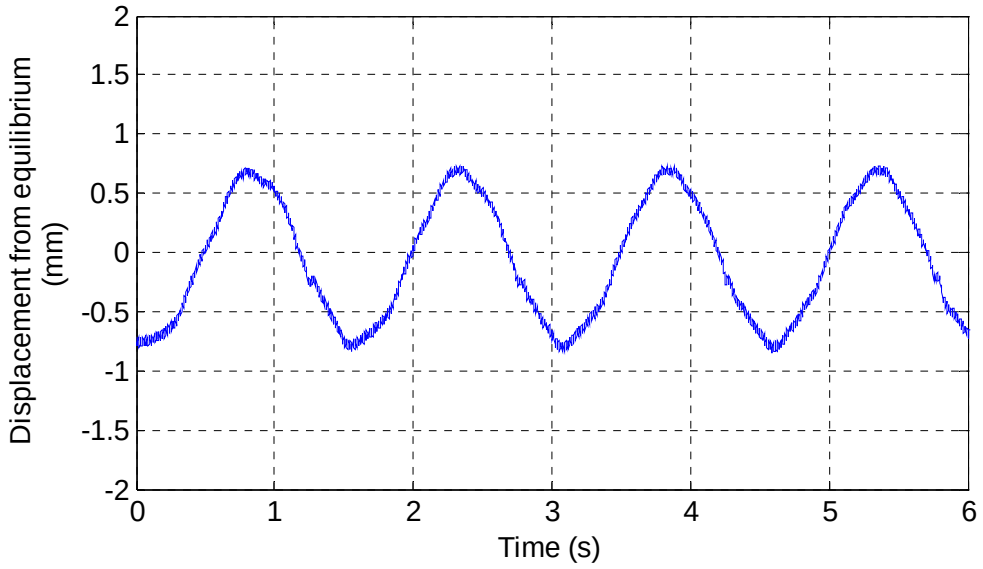


Fig. 2-9. Response of tracking a sinusoidal command of 0.7 Hz with a non-real-time operating system

Then we replaced the control system with Linux RTAI with Comedi. The conditions of the hardware setup were exactly the same as those in the first experiment. The data-acquisition board was again PCI-6025E. The PC used in this experiment contained the same processor (1.7-GHz Pentium IV) as the one used in the first experiment. Fig. 2-10 shows the system response of tracking a sinusoidal position command. The maximum command frequency that this NCS could follow was 2.8 Hz as shown in Fig.2-10. This factor-of-4 improvement in command frequency by comparing Figs. 2-9 and 2-10 resulted from the efficiency and the deterministic nature of our real-time operating environment.

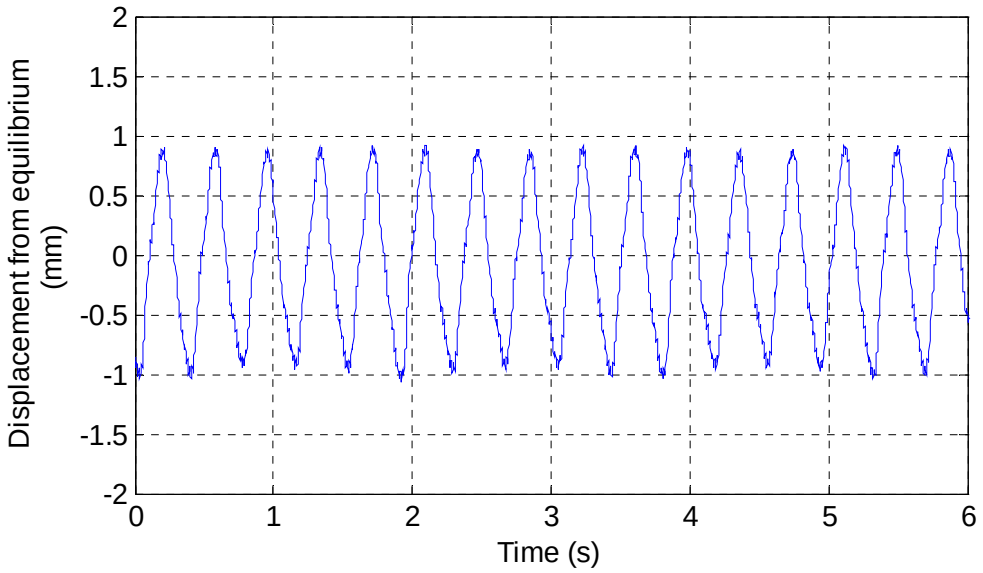


Fig. 2-10. Response of tracking a sinusoidal command with frequency of 2.8 Hz with the real-time operating environment

Several other experiments were performed to verify this real-time control over the Ethernet. Fig. 2-11 shows the system responses of tracking various ramp inputs and step inputs. Fig. 2-12 shows the system responses of tracking a sinusoidal command with different frequencies.

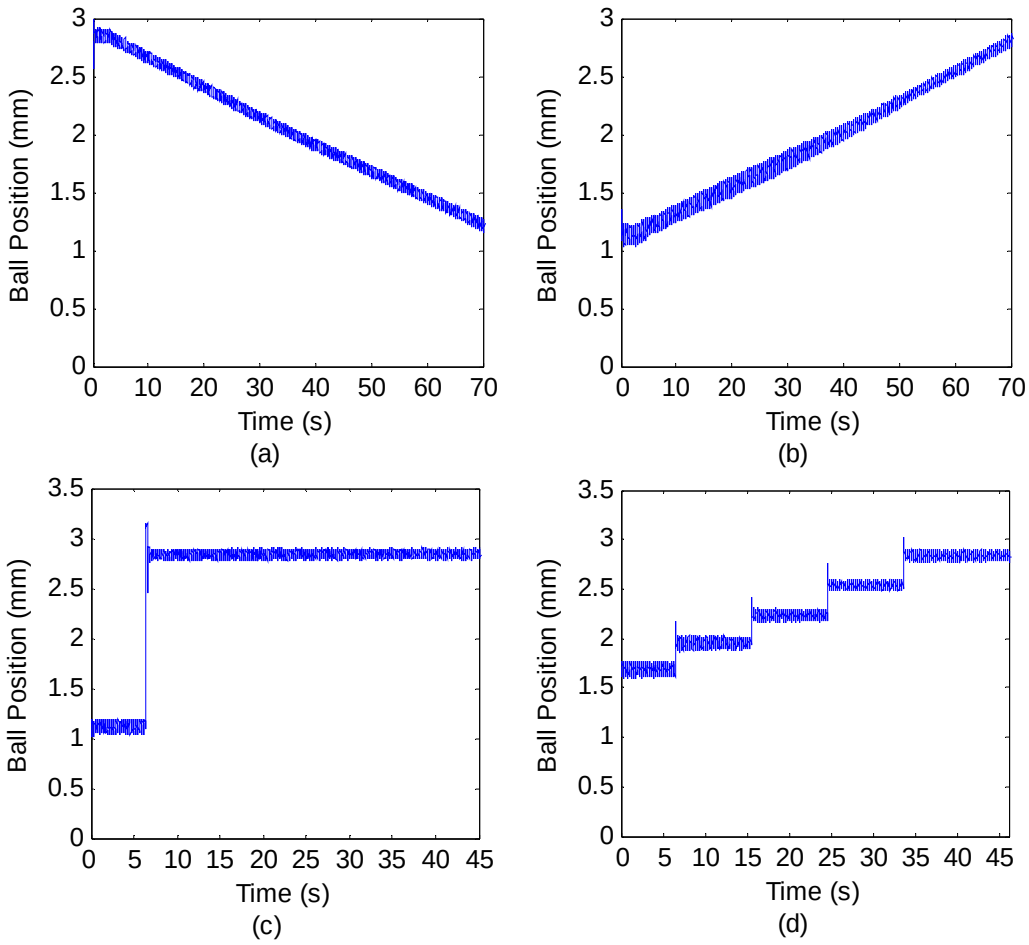


Fig. 2-11. Real-time system response of (a) tracking a decreasing ramp input, (b) tracking an increasing ramp input, (c) step input, and (d) multiple steps input

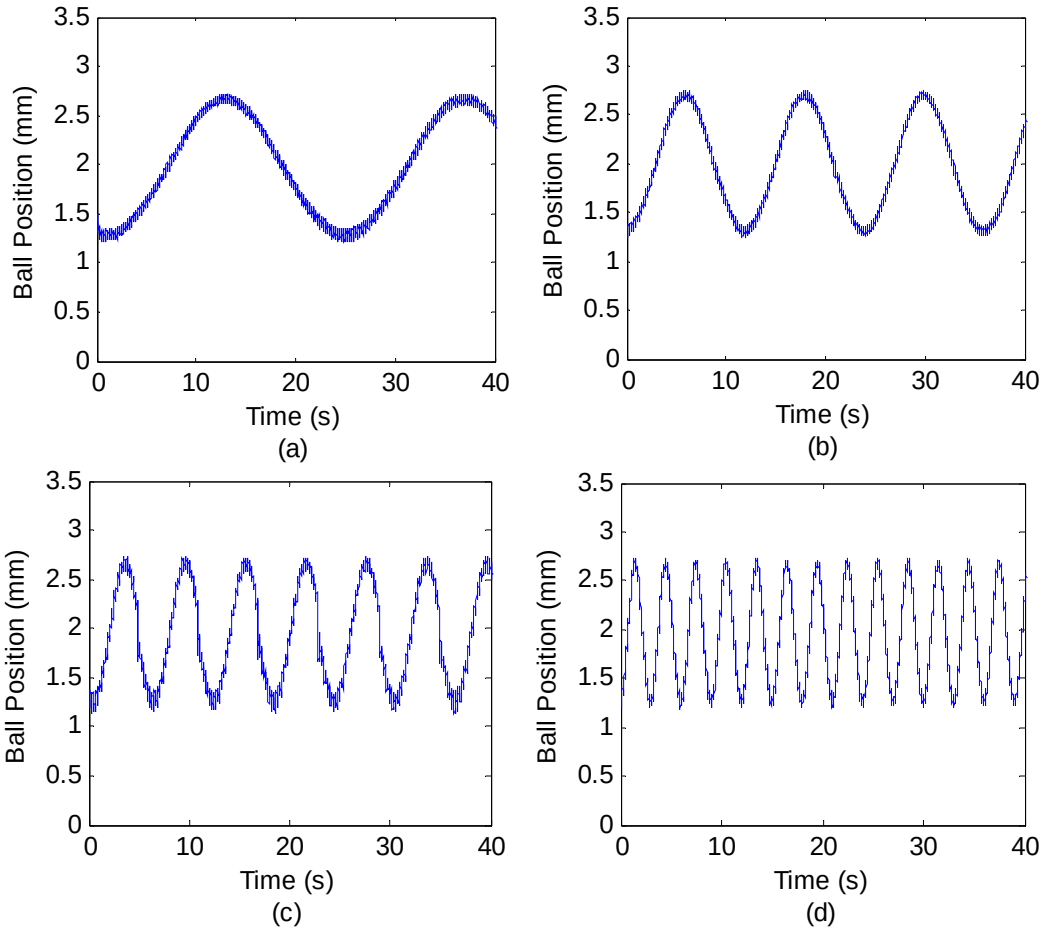


Fig. 2-12. System response of tracking a sinusoidal command with frequency of (a) 0.04 Hz, (b) 0.08 Hz, (c) 0.17 Hz, and (d) 0.33 Hz

2.4 Supervisory Control Based on the RTAI and PuTTY

Base on the real-time control environment proposed and implemented by Ambike [22] and a free secure shell client PuTTY, a simple Internet-based supervisory control of the ball maglev setup is implemented as shown in Fig. 2-13. Compared with the supervisory control introduced in Section 2.2, this approach is much simpler without any CGI programming.

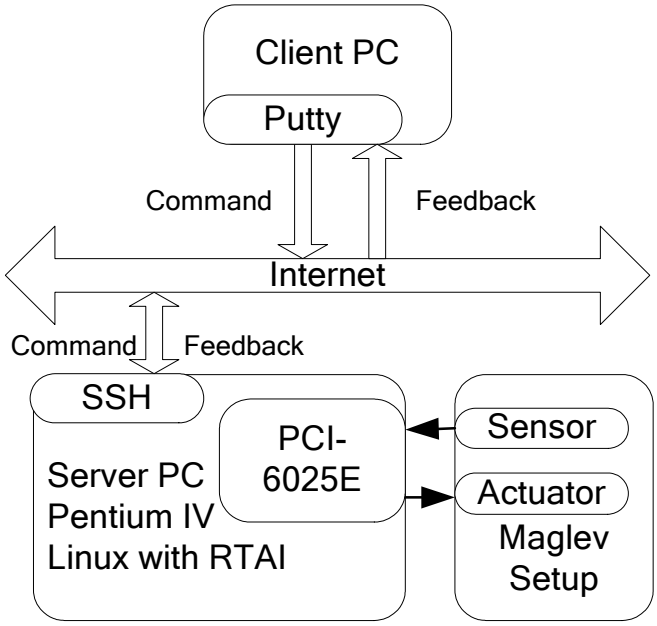


Fig. 2-13. Framework of the supervisory control of the test bed via the Internet based on RTAI and PuTTY

Multi-user operating systems, such as Unix and VMS, usually present a command-line interface to the user, much like the 'Command Prompt' or 'MS-DOS Prompt' in Windows. Using this type of interface, there is no need for user to be sitting at the same machine he is typing commands to. The commands, and responses, can be sent over a network, so the user can sit at one computer and give commands to another one, or even to more than one. SSH, Telnet and

Rlogin are network protocols that allow you to log in to a multi-user computer from another computer, over a network. On the computer the user sits at, he runs a client, which makes a network connection to the other computer (the server). The network connection carries the user's keystrokes and commands from the client to the server, and carries the server's responses back to the client. PuTTY is a free client of Telnet and SSH for Windows and Unix platforms [76].

When PuTTY is started, it shows a dialog box for configuration as shown in Fig. 2-14. In the 'Host Name' box is for the Internet host name or IP address of the server PC which controls the ball maglev setup. In the current research, the server PC installed with Linux and RTAI real-time system is named "maglev2".

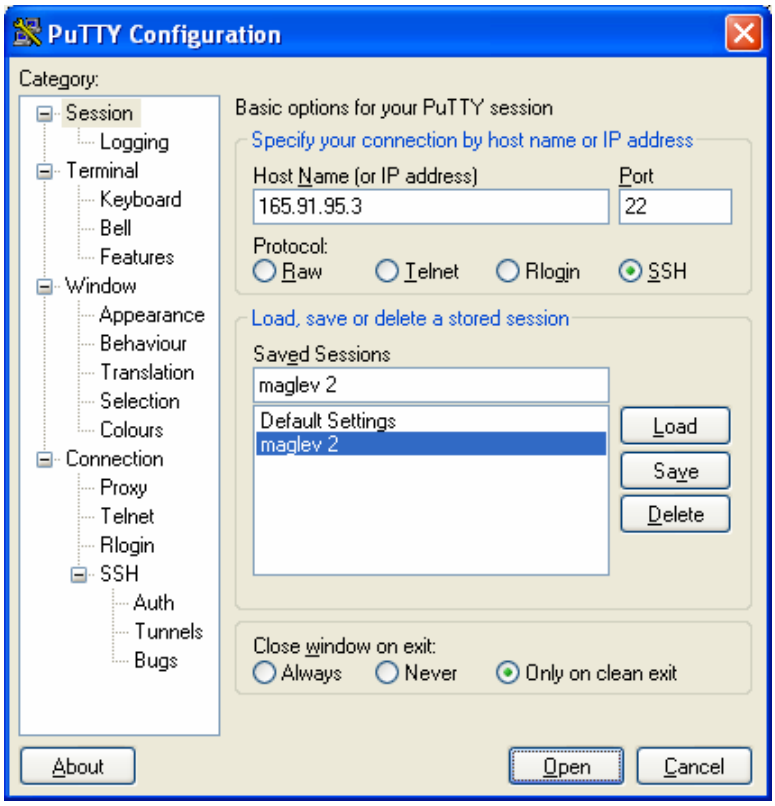


Fig. 2-14. Configuration window of PuTTY

Once the ‘Host Name’, ‘Protocol’, and possibly ‘Port’ settings are filled in and the ‘Open’ button is pressed, PuTTY will begin trying to connect to the server. If the connection is successful, a Command Prompt shows up, the server PC can be logged in with any client PC in the Internet with correct username and password. The control programs in the server PC can be run from the client PC and the control results can be seen from the client PC.

To verify the Internet-based supervisory control system implemented above, an experiment was performed. The plot of 300 μm step response obtained from the data in client PC side is shown in Fig. 2-15.

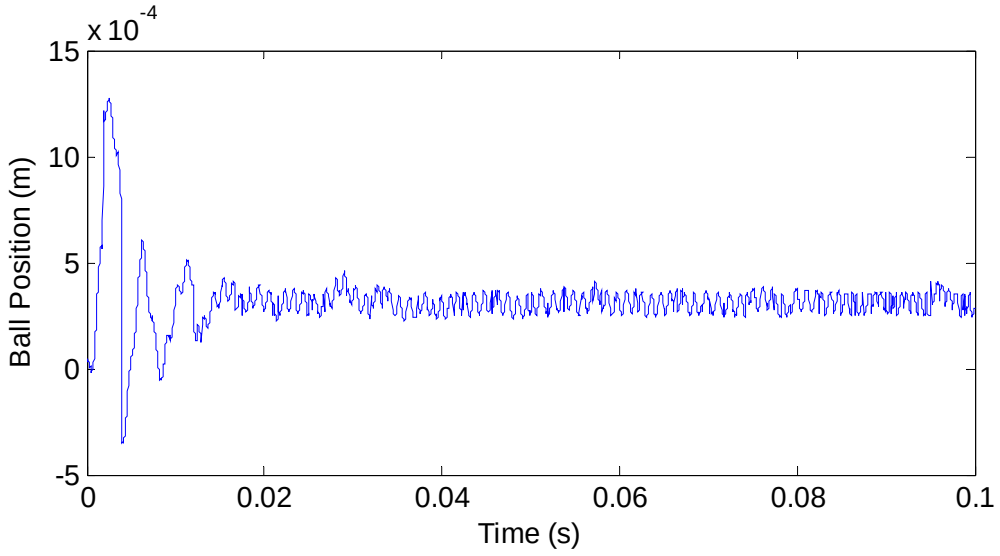


Fig. 2-15. Maglev system response to a step input of 300 μm provided via the Internet

2.5 Summary

In this chapter, the open-loop unstable ball maglev test bed was used to experimentally verify the feasibility of Internet-based real-time control. A real-time control over the Ethernet

was developed based on the real-time solution of RTAI with Linux proposed in [22]. UDP was used as the transport protocol due to its better real-time performance. The real-time operating environment improved the command-following capability by a factor of 4 in terms of command frequency. Thus feedback control over the Ethernet based on Linux with RTAI is demonstrated as one of the real-time solutions of real-time control over networks.

A new client/server-architecture-based supervisory control using the free SSH client PuTTY and the real-time operating environment is also proposed in this chapter. Experiment results based on real-time networked feedback control system and supervisory control system verified the feasibility of Internet-based real-time control. For supervisory control, system stability is not affected by the Internet since the control loop is closed locally. For real-time networked feedback control system implemented in this chapter, the system stability and performance are easily achieved since there are no significant time delays and data-packet losses in the LAN used in current research. However, for practical application, network-induced time-delay and packet-loss could be randomly time-varying. How to modeling system with time-delay and pack-loss and develop appropriate compensation algorithms to deal with them is further investigated in the following chapters.

CHAPTER III

MODELING AND STABILITY ANALYSIS OF NETWORKED FEEDBACK CONTROL

In this chapter, a new dynamic model of networked feedback control with the consideration of network-induced time delay, data-packet loss, and out-of-order data-packet transmission is provided.

3.1 Introduction

The recent migration of communication architectures from point-to-point to common-bus ones introduces different types of time delay uncertainty among sensors, actuators, and controllers. These time delays originate from the time-sharing nature of the communication medium as well as the computation time required for physical signal coding and communication processing. The characteristics of time delays could be constant, bounded, or even random, depending on the network protocols adopted and the chosen hardware. Input delay is the most common form of network-induced time delays or latencies in NCSs. It is well-known in control systems that time delays can degrade a system's performance and even cause system instability [78]. Considering the nondeterministic factors during data transmission, other network-communication induced phenomenon such as data-packet loss and out-of-order data transmission must also be addressed in deriving the continuous-time or discrete-time system model of a networked feedback control system.

3.2 Modeling of Feedback Control over Networks

Since NCSs are digital control systems with network communication, they usually contain continuous-time plants and discrete-time controllers with possibly different sampling frequencies. We consider a networked feedback control framework as shown in Fig. 3-1. Because of the limited network bandwidth, two classes of communication delays are included in this framework: (1) sensor-to-controller delay τ_{sc} and (2) controller-to-actuator delay τ_{ca}.

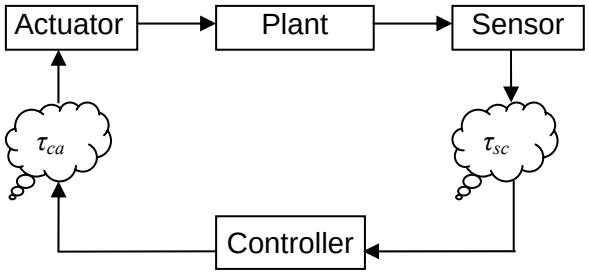


Fig. 3-1. Block diagram of a networked feedback control. The independent network delays from the controller to the actuator and from the sensor to the controller are denoted as τ_{ca} and τ_{sc}, respectively.

With the consideration of time delay and data-packet loss in data transmission, the system shown in Fig. 3-1 is described by the following dynamic model.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) + \mathbf{v}(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) + \mathbf{w}(t), \\ \mathbf{u}(t) &= \mathbf{u}(h_k), \qquad t \in \left[h_k + \tau_k, \quad h_{k+1} + \tau_{k+1}\right), \end{aligned} \tag{3.1}$$

where $\mathbf{x}(t) \in R^n$ is the state, $\mathbf{u}(t) \in R^m$ is the control input, $\mathbf{y}(t) \in R^q$ is the output, A, B, C, and D

are known real constant matrices of appropriate dimensions. With the sensor sampling period h, h_k denotes a certain sampling instant with $h_k = i_k h, \quad k = 0,\ 1,\ 2,\ \ldots$, where k is the index of the sensor sampling instants, and $i_k \in \{0,\ 1,\ 2,\ 3,\ \ldots\}$ is the index of the arrived packets at the actuator node. It is not required that $i_k < i_{k+1}$, which is discussed in Remarks 3-1 and 3-4 below, thus i_k can be different from k. The parameter τ_k denotes the time delay from the instant k when the sensor samples data to the instant when the actuator actuates the control input. We have $\tau_k = \tau_{sc} + \tau_{ca}$ for a fixed control law [10]. Let t_0 denote the instant when the control system is activated for the first time, then $\bigcup_{k=1}^{\infty}\left[h_k + \tau_k,\quad h_{k+1} + \tau_{k+1}\right) = \left[t_0,\quad \infty\right),\ t_0 \geq 0$. We define the following initial condition function

$$\mathbf{x}(t) = \boldsymbol{\theta}(t), \quad t \in \left[t_0 - \tau,\quad t_0\right). \tag{3.2}$$

In this chapter, the following assumptions are made:

3-1. The sensor node is time-driven and the actuator and controller nodes are event-driven.

3-2. The network-induced time delay is time varying with an unknown probability distribution function but bounded.

3-3. The plant noise $\mathbf{v}(t)$ and the sensor noise $\mathbf{w}(t)$ are zero-mean white Gaussian noises (WGNs) with $E\left\{\mathbf{v}^T(i)\mathbf{v}(j)\right\} = R_1\delta(i,j)$ and $E\left\{\mathbf{w}^T(i)\mathbf{w}(j)\right\} = R_1\delta(i,j)$, where $R_1 \geq 0, R_2 \geq 0$, and $\delta(i,j) = 1$ when $i = j,\ \delta(i,j) = 0$ when $i \neq j$. These noises are independent of previous states, control inputs, and network-induced time delays and packet losses.

3-4. There is no control input before the first control signal from the controller reaches the plant, i.e. $u(t) = 0$ for $t < t_0$.

Remark 3-1: $i_k \in \{0,\ 1,\ 2,\ 3,\ ...\}$, i.e. $\{i_0,\ i_1,\ i_2,\ i_3\ ...\}$ is a subset of $\{0,\ 1,\ 2,\ 3,\ ...\}$. From Assumption 3-2 the time delay is bounded. Then there exist constants $\tau > 0$ and $p > 0$ such that $\tau_k < h_{k+1} + \tau_{k+1} - h_k \le \tau = ph,\ k = 1,\ 2,\ \ldots$. There are several special cases in model (3.1–3.2):

1. If $i_{k+1} = i_k + 1$, then $\tau_k < h + \tau_{k+1}$. It includes two further special cases: (1) $\tau_k = \tau_0$, (the time delay is a constant); and (2) $\tau_k < h$, (the time delay is less than one sampling period).
2. If $\{i_0, i_1, i_2,\ ...,\ i_k\} = \{0,\ 1,\ 2,\ ...,\ k\}$, there is no data-packet loss in data transmission. Otherwise, missing integers indicate the lost data packets.
3. It is not required that $i_k < i_{k+1}$. If $i_k > i_{k+1}$, there is out-of-order data transmission between the i_k-th data packet and the i_{k+1}-th data packet.

Therefore, the system (3.1–3.2) represents a general form of the NCS model where the effects of the network-induced time delay, data-packet loss, and out-of-order data transmission are dealt with simultaneously.

Remark 3-2: The control input **u**(t) to the plant is piecewise constant during a sampling interval $[h_k,\ \ h_{k+1})$. From Assumptions 3-1 and 3-2 the actuator is event-driven, and from Remark 3-1, $\tau_k < ph,\ k = 1,\ 2,\ \ldots$. Thus there can be at most p + 1 control inputs in one sampling interval. A timing diagram of the data packets transmitted in the control loop is illustrated in Fig. 3-2. The solid arrows denote the control input delayed less than one sampling period. The dotted arrows denote that there are p + 1 delayed control inputs actuated at the actuator node in the sampling interval $[h_k,\ \ h_{k+1})$. The dashed arrows denote that there are 3 control inputs actuated at the actuator node in the sampling interval $[h_{k+3},\ \ h_{k+4})$. In the sampling interval $[h_k,\ \ h_{k+1})$, the control inputs arrive at the actuator node at the random instants $h_k + t_k^j$, where $0 \le t_k^j \le h, j \le p$.

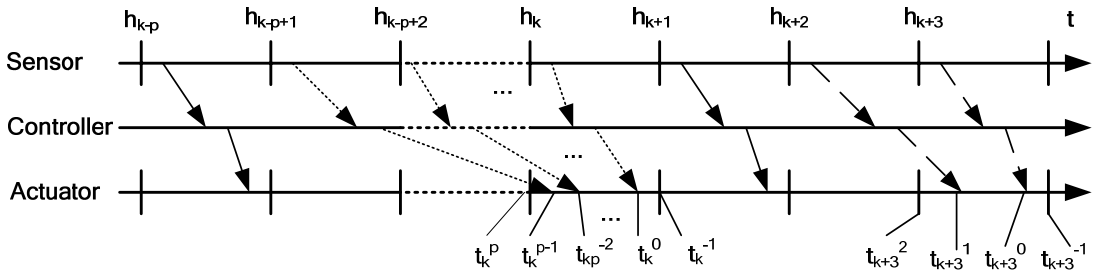


Fig. 3-2. Timing diagram of data packets transmitting in the control loop

In order to analyze the closed-loop system in the discrete-time domain, we use the following state-space solution of a set of first-order matrix differential equations to discretize the continuous-time plant dynamic model (3.1).

$$\mathbf{x}(t) = \exp(A(t-t_0))\mathbf{x}(t_0) + \int_{t_0}^{t} \exp(A(t-s))B\mathbf{u}(s)ds\ . \tag{3.3}$$

Substituting (3.3) to (3.1) with $t = h_{k+1}$ and $t_0 = h_k$ yields

$$\begin{aligned} \mathbf{x}(i_{k+1}) &= \Phi(h)\mathbf{x}(i_k) + \sum_{j=0}^{l} B_{i_k}^{j}(h)\mathbf{u}(i_{k-i}) + \mathbf{v}(i_k), \\ \mathbf{y}(i_k) &= C\mathbf{x}(i_k) + D\mathbf{u}(i_k) + \mathbf{w}(i_k), \qquad l = 1,\ 2,\ ...,\ p, \end{aligned} \tag{3.4}$$

where

$$\Phi(h) = e^{Ah}, B_{i_k}^{j}(h) = \int_{t_k^j}^{t_k^{j-1}} \exp(A(h-s))dsB, t_k^l = 0, t_k^{-1} = h, \mathbf{v}(i_k) = \int_{h_k}^{h_{k+1}} e^{A(h_{k+1}-s)}\mathbf{v}(s)ds,\ \ \mathbf{w}(i_k) = \mathbf{w}(h_k).$$

$\mathbf{v}(i_k)$ and $\mathbf{w}(i_k)$ are still zero-mean WGNs.

3.2.1 Full-State-Feedback Case

Full-state-feedback controller:

$$\mathbf{u}(t) = K\mathbf{x}(h_k), \quad t \in \left[h_k + \tau_k, \quad h_{k+1} + \tau_{k+1}\right) \tag{3.5}$$

where $K \in R^{m\times n}$ is a constant matrix.

3.2.2 Estimated-State-Feedback Case

Estimated-state-feedback controller:

$$\begin{aligned} \dot{\hat{\mathbf{x}}}(t) &= (A - LC)\hat{\mathbf{x}}(t) + \begin{bmatrix} B - LD & L \end{bmatrix} \begin{bmatrix} \mathbf{u}(t) \\ \mathbf{y}(t) \end{bmatrix} \\ \mathbf{u}(t) &= K\hat{\mathbf{x}}(h_k), \quad t \in \left[h_k + \tau_k, \quad h_{k+1} + \tau_{k+1}\right) , \end{aligned} \tag{3.6}$$

where $K \in R^{m\times n}$ and $L \in R^{m\times q}$ are constant matrices.

3.3 Sufficient Stability Condition

A controller design method for the system (3.1–3.2) with a full-state-feedback controller (3.5) is presented and a sufficient stability condition is derived based on a Lyapunov functional method in this section.

Lemma 3-1: For any vectors $\mathbf{u}, \mathbf{v} \in R^n$ and any real symmetric positive-definite matrix $P \in R^{n\times n}$, the following inequality holds.

$$-2\mathbf{u}^T\mathbf{v} \le \mathbf{u}^T P^{-1}\mathbf{u} + \mathbf{v}^T P\mathbf{v}\,. \tag{3.7}$$

Proof: Introduce the matrix $\mathbf{N} = P^{-\frac{1}{2}}\mathbf{u} + P^{\frac{1}{2}}\mathbf{v}$, and then we have

$$\mathbf{N}^T\mathbf{N} = (P^{-\frac{1}{2}}\mathbf{u} + P^{\frac{1}{2}}\mathbf{v})^T(P^{-\frac{1}{2}}\mathbf{u} + P^{\frac{1}{2}}\mathbf{v}) = \mathbf{u}^T P^{-1}\mathbf{u} + \mathbf{v}^T P\mathbf{v} + 2\mathbf{u}^T\mathbf{v} \ge 0.$$

Theorem 3-1: Given scalars $\tau > 0$ and $\lambda_i > 0\,(i = 2,\ 3,\ 4)$, the system (3.1–3.2) is closed-loop stable with a control input of $K = YX^{-T}$ if there exist real symmetric positive-definite matrices P and Q, a nonsingular matrix X, and matrices Y and Z_j (j = 1, 2, 3, 4) of appropriate dimensions such that the following LMI holds

$$\begin{bmatrix} M_{11}(X,Z_1) & M_{12}(X,Y,Z_1,Z_2,\lambda_2) & M_{13}(X,P,Z_3,\lambda_3) & M_{14}(Z_4,X) & M_{15}(Z_1,\tau) \\ * & M_{22}(Y,Z_2,\lambda_2) & M_{23}(X,Y,Z_3,\lambda_2,\lambda_3) & M_{24}(Z_4,X,\lambda_2) & M_{25}(Z_2,\tau) \\ * & * & M_{33}(X,Q,\lambda_3,\tau) & M_{34}(X,\lambda_3,\lambda_4) & M_{35}(Z_3,\tau) \\ * & * & * & M_{44}(X,\lambda_4) & M_{45}(Z_4,\tau) \\ * & * & * & * & M_{55}(Q,\tau) \end{bmatrix} < 0 \tag{3.8}$$

where

$$M_{11}(X,Z_1) = Z_1 + Z_1{}^T - AX^T - XA^T \tag{3.9}$$

$$M_{12}(X,Y,Z_1,Z_2,\lambda_2) = Z_2{}^T - Z_1 - \lambda_2 XA^T - BY \tag{3.10}$$

$$M_{13}(X,P,Z_3,\lambda_3) = Z_3{}^T - \lambda_3 XA^T + X^T + P \tag{3.11}$$

$$M_{14}(Z_4,X) = Z_4 - X^T - \lambda_4 XA^T \tag{3.12}$$

$$M_{15}(Z_1,\tau) = \tau Z_1 \tag{3.13}$$

$$M_{22}(Y,Z_2,\lambda_2) = -Z_2 - Z_2{}^T - \lambda_2(BY + Y^T B^T) \tag{3.14}$$

$$M_{23}(X,Y,Z_3,\lambda_2,\lambda_3) = -Z_3{}^T + \lambda_2 X^T - \lambda_3 Y^T B^T \tag{3.15}$$

$$M_{24}(Z_4, X, \lambda_2) = -Z_4 - \lambda_2 X^T - \lambda_4 Y^T B^T \tag{3.16}$$

$$M_{25}(Z_2, \tau) = \tau Z_2 \tag{3.17}$$

$$M_{33}(X, Q, \lambda_3, \tau) = \lambda_3 (X + X^T) + \tau Q \tag{3.18}$$

$$M_{34}(X, \lambda_3, \lambda_4) = (\lambda_4 - \lambda_3) X^T \tag{3.19}$$

$$M_{35}(Z_3, \tau) = \tau Z_3 \tag{3.20}$$

$$M_{44}(X, \lambda_4) = -2\lambda_4 X^T \tag{3.21}$$

$$M_{45}(Z_4, \tau) = \tau Z_4 \tag{3.22}$$

$$M_{55}(Q, \tau) = -\tau Q\ . \tag{3.23}$$

Proof: Consider the following Lyapunov function

$$V(t) = \mathbf{x}^T(t)\hat{P}\mathbf{x}(t) + \int_{t-\tau}^{t}\int_{v}^{t} \dot{\mathbf{x}}^T(s)\hat{Q}\dot{\mathbf{x}}(s)dsdv\ , \tag{3.24}$$

where $\hat{P}$ and $\hat{Q}$ are real symmetric positive definite matrices. With the following formulas

$$\int_{h_k}^{t} \dot{\mathbf{x}}(v)dv = \mathbf{x}(t) - \mathbf{x}(h_k) \tag{3.25}$$

$$\dot{X}(t) = \lim_{\delta \to 0^+} \sup \frac{X(t+\delta) - X(t)}{\delta} \tag{3.26}$$

$$\int_{t-\tau}^{t} \dot{\mathbf{x}}^T(v)\hat{Q}\dot{\mathbf{x}}(v)dv = \lim_{\delta \to 0^+}\ \delta \sum_{i=k-\frac{\tau}{\delta}}^{k} \left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right],\ t = k\delta \tag{3.27}$$

$$\int_{t-\tau}^{t}\int_{v}^{t} \dot{\mathbf{x}}^T(s)\hat{Q}\dot{\mathbf{x}}(s)dsdv = \lim_{\delta \to 0^+} \delta^2 \sum_{j=k-\frac{\tau}{\delta}}^{k} \sum_{i=j}^{k} \left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right] \tag{3.28}$$

and taking the time derivative of (3.24) for $t \in \left[h_k + \tau_k, \quad h_{k+1} + \tau_{k+1}\right)$ yields

$$\dot{V}(t) = 2\mathbf{x}^T(t)\hat{P}\dot{\mathbf{x}}(t) + \limsup_{\delta \to 0^+} \frac{\lim_{\delta \to 0^+}\left\{\delta^2 \sum_{j=k+1-\frac{\tau}{\delta}}^{k+1}\sum_{i=j}^{k+1}\left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right]\right\} - \lim_{\delta \to 0^+}\left\{\delta^2 \sum_{j=k-\frac{\tau}{\delta}}^{k}\sum_{i=j}^{k}\left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right]\right\}}{\delta}$$

$$= 2\mathbf{x}^T(t)\hat{P}\dot{\mathbf{x}}(t) + \lim_{\delta \to 0^+}\left\{\delta \sum_{j=k+1-\frac{\tau}{\delta}}^{k+1}\sum_{i=j}^{k+1}\left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right] - \delta \sum_{j=k-\frac{\tau}{\delta}}^{k}\sum_{i=j}^{k}\left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right]\right\}$$

$$= 2\mathbf{x}^T(t)\hat{P}\dot{\mathbf{x}}(t) + \lim_{\delta \to 0^+}\left\{\delta \frac{\tau}{\delta}\dot{\mathbf{x}}^T\left[(k+1)\delta\right]\hat{Q}\dot{\mathbf{x}}\left[(k+1)\delta\right]\right\} - \lim_{\delta \to 0^+}\left\{\delta \sum_{i=k-\frac{\tau}{\delta}}^{k}\left[\dot{\mathbf{x}}^T(i\delta)\hat{Q}\dot{\mathbf{x}}(i\delta)\right]\right\}$$

Applying (3.27) to above equation again, we obtain

$$\dot{V}(t) = 2\mathbf{x}^T(t)\hat{P}\dot{\mathbf{x}}(t) + \tau\dot{\mathbf{x}}^T(t)\hat{Q}\dot{\mathbf{x}}(t) - \int_{t-\tau}^{t}\dot{\mathbf{x}}^T(v)\hat{Q}\dot{\mathbf{x}}(v)dv. \tag{3.29}$$

From (3.1), (3.5), and (3.25), (3.29) can be rewritten as

$$\begin{aligned}\dot{V}(t) &= 2\mathbf{x}^T(t)\hat{P}\dot{\mathbf{x}}(t) + \tau\dot{\mathbf{x}}^T(t)\hat{Q}\dot{\mathbf{x}}(t) - \int_{t-\tau}^{t}\dot{\mathbf{x}}^T(v)\hat{Q}\dot{\mathbf{x}}(v)dv \\ &+2\left[\mathbf{x}^T(t)M_1 + \mathbf{x}^T(h_k)M_2 + \dot{\mathbf{x}}^T(t)M_3 + \mathbf{v}^T(t)M_4\right]\cdot\left[\mathbf{x}(t) - \mathbf{x}(h_k) - \int_{h_k}^{t}\dot{\mathbf{x}}(v)dv\right] \\ &+2\left[\mathbf{x}^T(t)N_1 + \mathbf{x}^T(h_k)N_2 + \dot{\mathbf{x}}^T(t)N_3 + \mathbf{v}^T(t)N_4\right]\cdot\left[\dot{\mathbf{x}}(t) - A\mathbf{x}(t) - BK\mathbf{x}(h_k) - \mathbf{v}(t)\right],\end{aligned} \tag{3.30}$$

where M_i and N_i (i = 1, 2, 3, 4) are arbitrary symmetric matrices of appropriate dimensions.

From Remark 3-1, we can assume that $\tau_k < h_{k+1} + \tau_{k+1} - h_k \le \tau$, then when $t \in \left[h_k + \tau_k, \quad h_{k+1} + \tau_{k+1}\right)$, we obtain $t - \tau \le h_k < t$. Thus

$$-\int_{t-\tau}^{t}\dot{\mathbf{x}}^T(v)\hat{Q}\dot{\mathbf{x}}(v)dv \le -\int_{h_k}^{t}\dot{\mathbf{x}}^T(v)\hat{Q}\dot{\mathbf{x}}(v)dv. \tag{3.31}$$

From Lemma 3-1, we obtain

$$-2\left[\mathbf{x}^T(t)M_1+\mathbf{x}^T(h_k)M_2+\dot{\mathbf{x}}^T(t)M_3+\mathbf{v}^T(t)M_4\right]\int_{h_k}^{t}\dot{\mathbf{x}}(v)dv\le\tau\tilde{\mathbf{x}}^T(t)\tilde{M}\hat{Q}^{-1}\tilde{M}^T\tilde{\mathbf{x}}(t)+\int_{h_k}^{t}\dot{\mathbf{x}}^T(v)\hat{Q}\dot{\mathbf{x}}(v)dv, \tag{3.32}$$

where $\tilde{\mathbf{x}}^T(t)=\left[\mathbf{x}^T(t)\quad\mathbf{x}^T(h_k)\quad\dot{\mathbf{x}}^T(t)\quad\mathbf{v}^T(t)\right]$ and $\tilde{M}=\left[M_1\quad M_2\quad M_3\quad M_4\right]^T$.

Combining (3.30–3.32), we obtain

$$\dot{V}(t)\le\tilde{\mathbf{x}}^T(t)\ \Lambda\ \tilde{\mathbf{x}}(t), \tag{3.33}$$

where

$$\Lambda=\begin{bmatrix} M_1+M_1^T-N_1A-A^TN_1^T & M_2^T-M_1-A^TN_2^T-N_1BK & M_3^T-A^TN_3^T+N_1+\hat{P} & M_4^T-N_1-A^TN_4^T \\ * & -M_2-M_2^T-N_2BK-K^TB^TN_2^T & -M_3^T+N_2-K^TB^TN_3^T & -M_4^T-N_2-K^TB^TN_4^T \\ * & * & N_3+N_3^T+\tau\hat{Q} & N_4^T-N_3 \\ * & * & * & -N_4-N_4^T \end{bmatrix}+\tau\tilde{M}\hat{Q}^{-1}\tilde{M}^T.$$

Applying Schur complements [77], the condition $\Lambda<0$ is equivalent to the following LMI.

$$\begin{bmatrix} M_1+M_1^T-N_1A-A^TN_1^T & M_2^T-M_1-A^TN_2-N_1BK & M_3^T-A^TN_3^T+N_1+\hat{P} & M_4-N_1-A^TN_4^T & \tau M_1 \\ * & -M_2-M_2^T-N_2BK-K^TB^TN_2^T & -M_3^T+N_2-K^TB^TN_3^T & -M_4-N_2-K^TB^TN_4^T & \tau M_2 \\ * & * & N_3+N_3^T+\tau\hat{Q} & N_4^T-N_3 & \tau M_3 \\ * & * & * & -N_4-N_4^T & \tau M_4 \\ * & * & * & * & -\tau\hat{Q} \end{bmatrix}<0. \tag{3.34}$$

Define $N=N_1, N_2=\lambda_2 N, N_3=\lambda_3 N, N_4=\lambda_4 N, X=N^{-1}, Y=KX^T, Z_i=XM_iX^T, P=X\hat{P}X^T$, and $Q=X\hat{Q}X^T$, then pre- and post-multiplying both sides of (3.34) with diag($X\ X\ X\ X\ X$) and its transpose, respectively, we can show that (3.34) is equivalent to (3.8) and complete the proof.

Remark 3-3: The above Lyapunov method was applied in the continuous-time domain, so the inter-sample behavior was taken into account. Moreover, the network-induced delay considered here includes τ_{sc} and τ_{ca} and could be time-varying.

Remark 3-4: From Remark 3-1, if $h_k > h_{k+1}$, there is out-of-order data transmission between the i_k-th and i_{k+1}-th data packets. That is, the new data packet containing $\mathbf{x}(h_k)$ reaches the plant before the old one containing $\mathbf{x}(h_{k+1})$. Appropriately discarding the outdated data packet can save network bandwidth and thus reduce the networked-induced time delay, which in turn makes the system be able to tolerate a larger amount of data-packet loss. Therefore, it is necessary to time-stamp the data and establish an appropriate network scheduling method at the sensor node that can discard the outdated untransmitted messages when the new packet is transmitted. The controller may also discard the outdated sensor data to reduce the network traffic between the controller node and the actuator node.

To verify the sufficient stability condition derived above, consider the following system:

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}\mathbf{u}(t) + \mathbf{v}(t), \\
\mathbf{y}(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix}\mathbf{x}(t) + \mathbf{w}(t), \\
\mathbf{u}(t) &= K\mathbf{x}(t).
\end{aligned} \tag{3.35}$$

This example is the same as Example 1 in [28] except that we introduced noises $\mathbf{v}(t)$ and $\mathbf{w}(t)$. From [28], the maximum allowable time delay that does not affect the stability of the system is 0.0538 s. By using Theorem 3-1 and solving the LMI (3.8), the upper bound of time delay τ for (3.35) is found to be 0.964 s and the feedback gain $K = \begin{bmatrix} -3.75 & -11.5 \end{bmatrix}$. A step response of the system with time delays bounded by 0.964 s and $R_1 = 0$, $R_2 = 0.001$ is shown in Fig. 3-3. Thus, the sufficient stability condition in Theorem 3-1 is less conservative than the one given in [28].

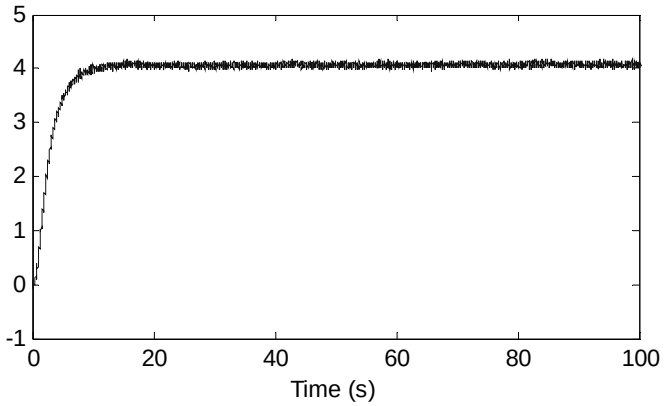


Fig. 3-3. Step response with time delays bounded by 0.964 s

3.4 Summary

A new dynamic model for NCSs with the consideration of network-induced time delay, data-packet loss, and out-of-order data-packet transmission was provided in this chapter. Based on this model, the methods of stability and controller design were proposed. A new delay-dependent stability criterion and the upper bound of the time delays that the system can accommodate were derived through a Lyapunov functional approach by solving a LMI. Furthermore, the relation between the sampling period and the network-induced time delay can also be determined based on this approach, which will be discussed in Chapter V. The appropriate co-design integration of control systems, real-time systems, and network communication systems proposed in this dissertation is based on this model and will be presented in the following chapters.

CHAPTER IV

TIME-DELAY AND PACKET-LOSS COMPENSATION

In Chapter II we discussed the design and implementation of a real-time control of the ball maglev system via an Ethernet LAN. This chapter describes time-delay and packet-loss compensation algorithms to ensure the system stability of the ball maglev test bed in the presence of sporadic delays and successive data-packet losses. Two approaches based on two different combinations of timing schemes are proposed in detail.

4.1 Introduction

Randomly-varying time delays and data-packet losses induced by the network are well known to degrade the system stability and performance [7] [28–29]. To successfully implement the real-time networked control of the ball maglev system developed in Chapter II, we need to collect the delay information of the Ethernet LAN used as the communication medium. We implement the real-time control of this ball maglev test bed through Ethernet LANs.

With a server PC and a client PC connected to the LAN in our lab, the round-trip time delay was measured with the c programming code provided by Ambike. The time delay profile is plotted and shown in Fig. 4-1. We repeated this delay measurement many times and the average round trip time delay between the client PC and the server PC was found to be about 230 μs and with standard deviation of about 200 μs. But there are some sporadic delays as long as 3.5 ms which is on the order of the system sampling period of 3 ms as shown in Fig. 4-1.

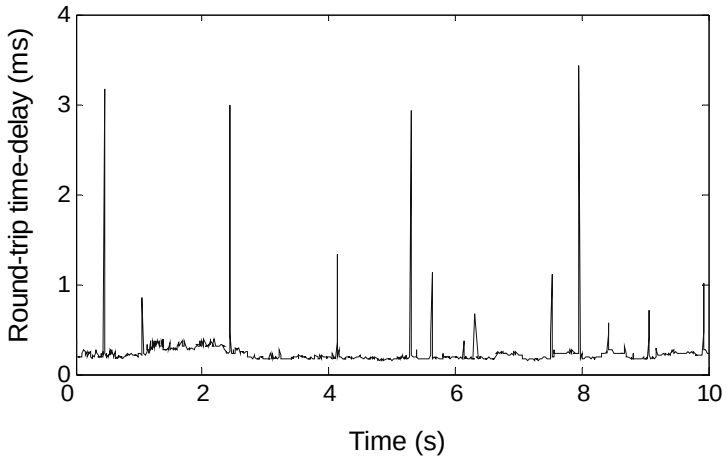


Fig. 4-1. Profile for round-trip time delays between two PCs connected to a LAN

With a server PC and a client PC connected to two separate LANs on the Texas A&M University campus, experimental delay data are collected and shown in Fig. 4-2. As in a typical network, sporadic surges in time delays were observed in our LANs shown in a delay profile in Fig. 4-2.

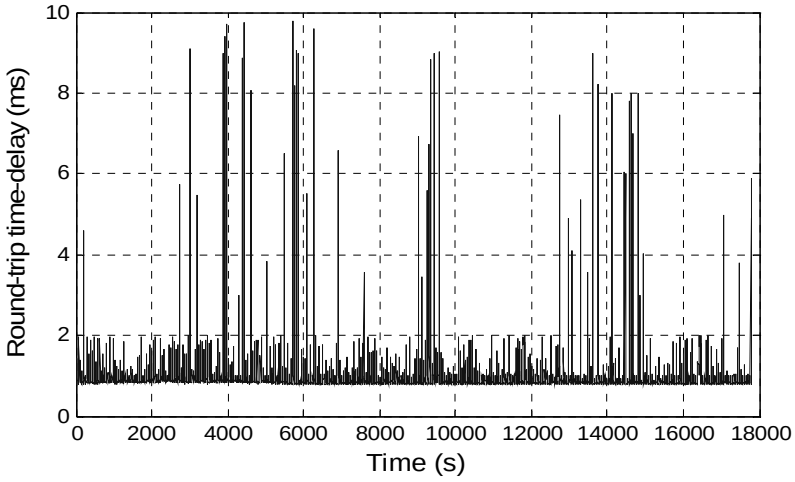


Fig. 4-2. Profile for round-trip time delays between two PCs connected to two separate LANs

There are two timing schemes: clock driven and event driven. The various timing methods used for sensor sensing, controller calculating, and actuator actuating have different implications for node synchronization. A summary of the different timing scheme combinations used is presented in Table 4-1 [10].

Table 4-1. Types of timing schemes of control nodes [10]

Sensor	Controller	Actuator	Clock synchronization
Clock	Clock	Clock	Yes
Clock	Clock	Event	Yes
Clock	Event	Event	No
Clock	Event	Clock	No

In the past, synchronization of clocks was tried to coordinate the events in the networked control loop [79]. It is a complicated process generating additional network traffic to deliver the synchronized clock signals, and also requires continuous adjustment to retain this synchronization once generated. Two time-delay/data-loss compensation approaches based on the two timing schemes in which no time synchronization is required are proposed in this chapter.

For the networked feedback control system shown in Fig. 3-1, the system stability could be lost because of the presence of bounded sporadic surges in communication time delays in the LANs in our labs shown in Fig. 4-2. The two classes of time delays and packet losses all need to be dealt with. The delay τ_{sc} can be determined by the controller node if the sensor clock and the controller clock are synchronized and all the sensor data packets are time-stamped. The delay τ_{ca} is different in nature because the controller node does not know how long it will take for its control signal to reach the actuator node and no correction can be made at the controller node during the control calculation [28]. Thus we need to deal with these two classes of time delays separately and the packet losses as well.

4.2 Model Estimation-Based Approach

The first approach is based on a combination of time-driven sensor and event-driven controller and actuator, and a multi-step-ahead model estimation. Since the network-induced delays and packet losses are random with unknown distribution and the upper bound of time delay is less than p sampling intervals from Assumption 3-1, the discrete-time model of the NCS (3.4) is varying with the length of the delay and the rate of the packet losses. Denote l sampling intervals as the bound of the varying length of the time delay at a given instant of time and l = 1, 2, 3, ···, or p. The discrete-time model of the system with the sampling period h and round-trip delay τ_k can be rewritten as

$$\begin{aligned}
&\mathbf{x}(i_{k+1}) = \Phi\mathbf{x}(i_k) + \Gamma\mathbf{U}(i_{K-1}) + \mathbf{v}(i_k)\\
&\cdots\\
&\mathbf{x}(i_{k+l-1}) = \Phi\mathbf{x}(i_{k+l-2}) + \Gamma\mathbf{U}(i_{K-1}) + \mathbf{v}(i_{k+l-2})\\
&\mathbf{x}(i_{k+l}) = \Phi\mathbf{x}(i_{k+l-1}) + \Gamma_0(\tau_k)\mathbf{U}(i_K) + \Gamma_1(\tau_k)\mathbf{U}(i_{K-1}) + \mathbf{v}(i_{k+l-1})\\
&\mathbf{y}(i_k) = C\mathbf{x}(i_k) + D\mathbf{U}(i_K) + \mathbf{w}(i_k)
\end{aligned} \tag{4.1}$$

where $\Phi = e^{Ah}, \Gamma_0(\tau_k) = \int_0^{lh-\tau_k} e^{As}Bds, \Gamma_1(\tau_k) = \int_{lh-\tau_k}^{h} e^{As}Bds, \Gamma = \int_0^h e^{As}Bds$ and $\tau_k < lh$. The time-driven index i_k is the index of sampling-time instant as defined with $h_k = i_k h, k = 0, 1, 2, \ldots$ in Section 3.2. The parameter K denotes the event number. The index i_K is the index of the time instant when the sensor data arrive at the controller node and the updated control signal is available, so it is dependent on the time delay τ_{sc} and the rate of packet losses between the sensor node and the controller node. The relation between $\mathbf{U}(i_K)$ and $\mathbf{u}(i_k)$ will be established in the following section.

4.2.1 Compensation Algorithm for Time Delay and Packet Loss

We know that the worst case is that the control signal does not arrive at the actuator node in p sampling intervals. This could be due to the time delay that is up to p sampling-period long or $p - 1$ consecutive packet losses between the controller node and the actuator node. Thus we design an estimator at the controller node to estimate the plant states of the successive samples p steps in advance. With these estimated states the controller calculates the control signals for each of the following p sampling intervals then sends them as a package to the actuator node. The actuator node then adopts the corresponding control signal from the package in the current sampling interval.

The p-step-ahead state estimation is done as follows

$$\begin{aligned}
&\hat{\mathbf{x}}(i_{k+1}) = \Phi\mathbf{x}(i_k) + \Gamma\mathbf{u}(i_k) \\
&\hat{\mathbf{x}}(i_{k+2}) = \Phi\hat{\mathbf{x}}(i_{k+1}) + \Gamma\mathbf{u}(i_{k+1}) \\
&\ldots \\
&\hat{\mathbf{x}}(i_{k+p}) = \Phi\hat{\mathbf{x}}(i_{k+p-1}) + \Gamma\mathbf{u}(i_{k+p-1}) \ .
\end{aligned} \tag{4.2}$$

The control signal package is generated as

$$\begin{aligned}
&\mathbf{u}(i_k) = -L\mathbf{x}(i_k) \\
&\mathbf{u}(i_{k+1}) = -L\hat{\mathbf{x}}(i_{k+1}) \\
&\ldots \\
&\mathbf{u}(i_{k+p-1}) = -L\hat{\mathbf{x}}(i_{k+p-1}) \ ,
\end{aligned} \tag{4.3}$$

where L denotes the control law without assuming delay. Thus each control signal package transmitted to the actuator node includes $\mathbf{u}(i_k), \mathbf{u}(i_{k+1}), \ \ldots \ , \mathbf{u}(i_{k+p-1})$.

The actuator node chooses the control signal from the package as below for the next p sampling intervals until the new control signal package arrives

$$\mathbf{U}(i_K) = \begin{cases} \mathbf{u}(i_k) \text{ for } h_k \le t < h_{k+1} \\ \mathbf{u}(i_{k+1}) \text{ for } h_{k+1} \le t < h_{k+2} \\ \ldots \\ \mathbf{u}(i_{k+p-1}) \text{ for } h_{k+p-1} \le t < h_{k+p} \end{cases},$$

where $\mathbf{U}(i_K)$ denotes the actual control signal adopted by the actuator, $\mathbf{u}(i_k), \mathbf{u}(i_{k+1}), \ldots,$ $\mathbf{u}(i_{k+p-1})$ denote the components of the incoming control signal package at the corresponding sampling time interval i_k, and t denotes the continuous time. Once the new control signal package arrives, (4.4) is revised with the new control signal components, and $\mathbf{U}(i_{K+1})$ is then available. An actual real-time C implementation to select proper control signal from the control signal package at the actuator node is

$$\begin{aligned} \mathbf{U}(i_K) &= 1/4 \ (1 - \text{sign}(t - h_{k+1}))(1 + \text{sign}(t - h_k)) \ \mathbf{u}(i_k) \\ &+ 1/4 \ (1 - \text{sign}(t - h_{k+2}))(1 + \text{sign}(t - h_{k+1})) \ \mathbf{u}(i_{k+1}) \\ &+ 1/4 \ (1 - \text{sign}(t - h_{k+3}))(1 + \text{sign}(t - h_{k+2})) \ \mathbf{u}(i_{k+2}) \\ &+ \ldots \\ &+ 1/4 \ (1 - \text{sign}(t - h_{k+p}))(1 + \text{sign}(t - h_{k+p-1})) \ \mathbf{u}(i_{k+p-1}). \end{aligned} \tag{4.4}$$

where $\text{sign}(x) = \begin{cases} 1 & x \ge 0 \\ -1 & x < 0 \end{cases}$. Expanding (4.4), we have the following expression that relates the actual control signal $\mathbf{U}(i_K)$ adopted by the actuator node and the individual control signal components $\mathbf{u}(i_k)$ in the control signal package.

$$
\begin{aligned}
&\text{if } \tau_k \le h, \text{ then}\\
&\mathbf{U}(i_K) = \mathbf{u}(i_k) \text{ for } h_k \le t < h_{k+1};\\
&\mathbf{U}(i_{K+1}) = \mathbf{u}(i_{k+1}) \text{ for } h_{k+1} \le t < h_{k+2};\\
&\ldots\\
&\text{if } h \le \tau_k < 2h, \text{ then}\\
&\mathbf{U}(i_K) = \mathbf{u}(i_k) \text{ for } h_k \le t < h_{k+1};\\
&\mathbf{U}(i_K) = \mathbf{u}(i_{k+1}) \text{ for } h_{k+1} \le t < h_{k+2};\\
&\mathbf{U}(i_{K+1}) = \mathbf{u}(i_{k+2}) \text{ for } h_{k+2} \le t < h_{k+3};\\
&\ldots\\
&\text{if } h_{p-1} \le \tau_k < h_p, \text{ then}\\
&\mathbf{U}(i_K) = \mathbf{u}(i_k) \text{ for } h_k \le t < h_{k+1};\\
&\ldots\\
&\mathbf{U}(i_K) = \mathbf{u}(i_{k+p-1}) \text{ for } h_{k+p-1} \le t < h_{k+p};\\
&\mathbf{U}(i_{K+1}) = \mathbf{u}(i_{k+p}) \text{ for } h_{k+p} \le t < h_{k+p+1};\\
&\ldots
\end{aligned}
\tag{4-5}
$$

The denotation $\mathbf{U}(i_K)$ is unchanged but its value is varying during the time delay and then updated to be $\mathbf{U}(i_{K+1})$ when a new control signal package is available. If the time delay is less than h, there is an updated control signal available during two successive sampling intervals. If time delay is less than 2 h, there is an updated control signal available during three successive sampling intervals, and so on. In case the new control signal package does not arrive, the actuator node can use formerly calculated and stored control signals for up to the next $p - 1$ sampling intervals from the control signal package that arrived most recently. The compensation for time delays and packet losses simultaneously with the algorithm developed above is elaborated in the following.

For example, with $p = 4$, how the control signals are adopted by the actuator corresponding to the sampling intervals are illustrated in Fig. 4-3 when different time delays or packet losses occur. Note again that i_K is the event index while i_k is the discrete-time index.

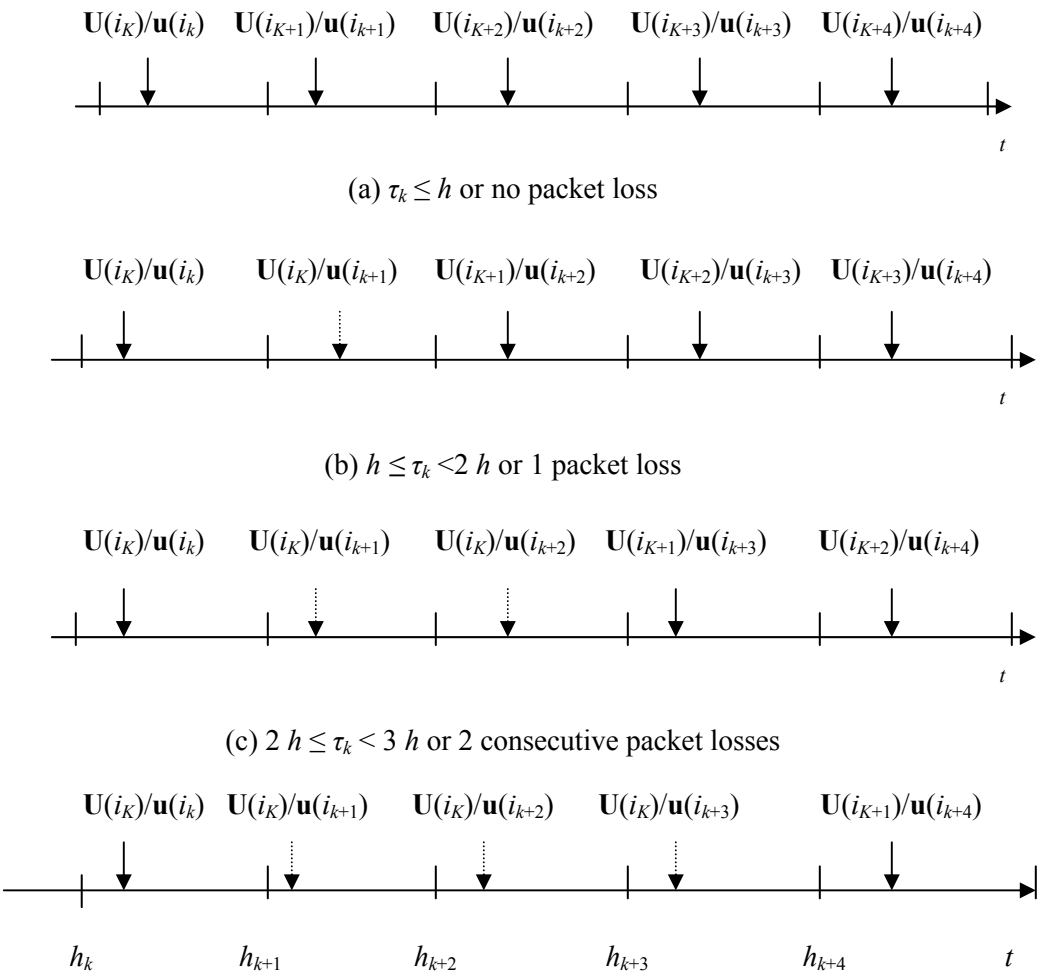


Fig. 4-3. Actual control signal $\mathbf{U}(i_K)$ adopted by the actuator node and components of the control signal packet $\mathbf{u}(i_k)$ with respect to time when different time delays or packet losses occur. In this figure p is assumed to be 4, and parts (a), (b), (c) and (d) cover all the possible cases of the time delay up to 4 h or the number of consecutive packet losses up to 3. Short vertical lines indicate sampling instants. Solid arrows indicate that new control signal package arrive in the corresponding sampling interval. Dotted arrows indicate that the incoming control signal package is delayed or lost.

If there is no new control signal package available in any given sampling interval denoted as a dotted arrow in Fig. 4-3, the formerly estimated control signal by (4.5) and stored in the last available control signal package is used. If a new control signal package arrives in any given sampling interval denoted as a solid arrow in Fig. 4-3, it revises all the components of the

last available control signal package. Then $\mathbf{U}(i_{K+1})$ is now available. Comparing $\mathbf{U}(i_{K+1})$, $\mathbf{U}(i_{K+2})$, $\mathbf{U}(i_{K+3})$, and $\mathbf{U}(i_{K+4})$ with the definition in (4-5), the actuator receives the proper control signal in each sampling interval regardless of time delays or packet losses. In case of an out-of-order transmission and arrival of packages, the outdated packages are simply discarded. Fig. 4-4 shows an example communication process that includes all the situations of time delays, data-packet losses, and out-of-order data transmissions. Our algorithm works well when these situations occur simultaneously. The labels in bold face such as $\boldsymbol{u_{2e}}$, $\boldsymbol{u_{3e}}$, and so on denote the estimated control signals. The labels in unbold face such as u_1, u_5, and u_7 denote the real control signals. In Fig. 4-4, the numerical indices of the control signals the plant receives is exactly the same as those of the sampling intervals, thus in every sampling interval, the plant receives a proper control signal that is either real or most recently estimated in the previous sampling intervals. Thus this algorithm compensates for time delays, packet losses, and out-of-order packet transmissions in a unified way.

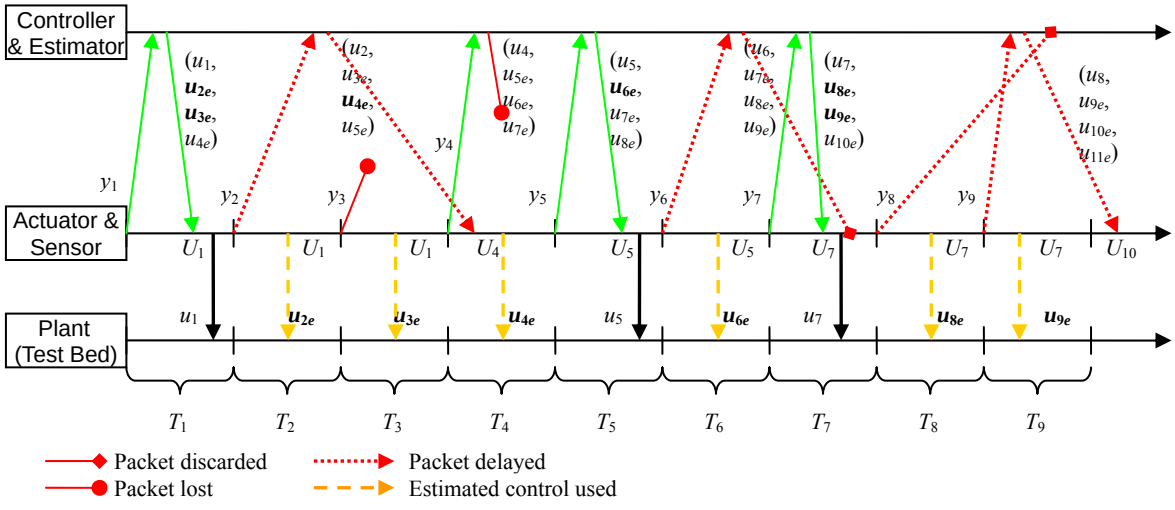


Fig. 4-4. Example communication process with time delays, packet losses, and out-of-order packet arrivals altogether

4.2.2 Sufficient Stability Conditions for Compensated Systems

With the model-estimation-based compensation algorithm developed above, the plant receives the control input at each time instant k. Thus we can develop augmented system equations to facilitate the stability conditions. We consider the full-state-feedback case shown in Fig. 4-5 and the output-feedback case shown in Fig. 4-6.

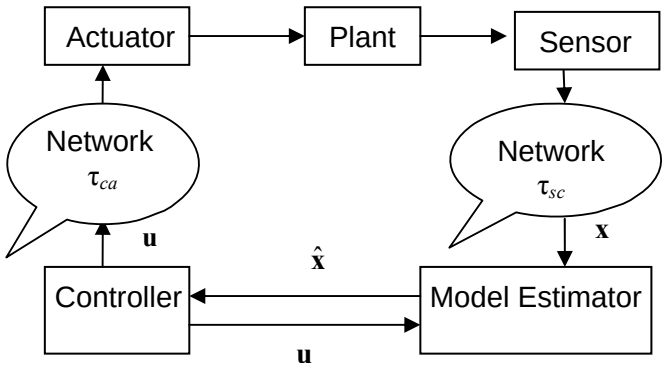


Fig. 4-5. Block diagram of a full-state-feedback NCS with a model estimator

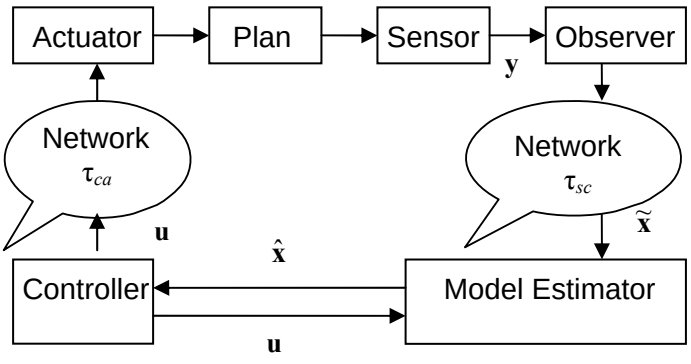


Fig. 4-6. Block diagram of an output-feedback NCS with a model estimator

If all the states are measurable, the sensor mode can send the full states through the network to update the model estimator as shown in Fig. 4-5. Assume that the sampling period is *h*. Then we have following difference equations:

Discrete-time plant model:

$$\begin{aligned}\mathbf{x}(i_{k+1}) &= \Phi\mathbf{x}(i_k) + \Gamma\mathbf{u}(i_k) + \mathbf{v}(i_k)\\ \mathbf{y}(i_k) &= C\mathbf{x}(i_k) + D\mathbf{u}(i_k) + \mathbf{w}(i_k)\end{aligned} \tag{4.6}$$

Model estimator:

$$\begin{aligned}\hat{\mathbf{x}}(i_{k+1}) &= \Phi\hat{\mathbf{x}}(i_k) + \Gamma\mathbf{u}(i_k) + \mathbf{v}(i_k)\\ \hat{\mathbf{y}}(i_k) &= C\hat{\mathbf{x}}(i_k) + D\mathbf{u}(i_k) + \mathbf{w}(i_k)\end{aligned} \tag{4.7}$$

Full-state-feedback controller:

$$\mathbf{u}(i_k) = L\hat{\mathbf{x}}(i_k) \tag{4.8}$$

State-estimation error:

$$\mathbf{e}(i_k) = \mathbf{x}(i_k) - \hat{\mathbf{x}}(i_k),\ i_k \in \left[i_K, i_{K+1}\right). \tag{4.9}$$

We define a positive integer variable $N(K) = i_{K+1} - i_K$, where i_k is the index of sampling instants defined in Chapter III, and i_K is the index of event-arrival instants. Thus,

$$\mathbf{e}(i_K) = \mathbf{x}(i_K) - \hat{\mathbf{x}}(i_K) = 0. \tag{4.10}$$

In other words, the state error in the model estimator is reset to 0 at the i_K-th instant when the actual sensor data are available.

When it is impossible to directly measure all the plant states, a state observer may be implemented as shown in Fig. 4-6. This observer sends the observed states to the model estimator. Similarly, with the same dynamic equations (4.6–4.8) for the observer,

$$\tilde{\mathbf{x}}(i_{k+1}) = (\Phi - L_o C)\tilde{\mathbf{x}}(i_k) + \begin{bmatrix} \Gamma - L_o D & L_o \end{bmatrix} \begin{bmatrix} \mathbf{u}(i_k) \\ \mathbf{y}(i_k) \end{bmatrix}, \tag{4.11}$$

where $\tilde{\mathbf{x}}(i_k)$ is the observer state vector, and L_o is the observer gain. Define

$$\bar{\mathbf{e}}(i_k) = \begin{cases} \tilde{\mathbf{x}}(i_k) - \hat{\mathbf{x}}(i_k), & i_k \in (i_K, i_{K+1}) \\ 0, & i_k = i_K \end{cases}. \tag{4.12}$$

Remark 4-1: The NCS model in [27] is extended to be the NCS framework shown in Fig. 3-2. This can deal with the case where there are time delays and data-packet losses in both the feedback path and the forward path whereas only the time delay in the feedback path was considered in [27]. The algorithm developed in the previous section can compensate for these two classes of time delays and data-packet losses simultaneously.

1. For the full-state-feedback NCS shown in Fig. 4-5, the augmented system equation is

$$\begin{bmatrix} \mathbf{x}(i_{k+1}) \\ \mathbf{e}(i_{k+1}) \end{bmatrix} = \begin{bmatrix} \Phi + \Gamma L & -\Gamma L \\ 0 & \Phi \end{bmatrix} \begin{bmatrix} \mathbf{x}(i_k) \\ \mathbf{e}(i_k) \end{bmatrix}. \tag{4.13}$$

Define $\mathbf{z}_1(i_k)=\begin{bmatrix}\mathbf{x}(i_k)\\ \mathbf{e}(i_k)\end{bmatrix}$, $\Lambda_1=\begin{bmatrix}\Phi+\Gamma L & -\Gamma L\\ 0 & \Phi\end{bmatrix}$, then (4.13) can be rewritten as

$$\mathbf{z}_1(i_{k+1})=\Lambda_1\mathbf{z}_1(i_k),\quad i_k\in[i_K,i_{K+1}). \tag{4.14}$$

The system described by (4.14) with the initial conditions $\mathbf{z}_0=[\mathbf{x}(i_0)\quad 0]^T$ has the following solution [27].

$$\mathbf{z}_1(i_k)=\Lambda_1^{\,i_k-i_K}\prod_{i=0}^{K}\left(\begin{bmatrix}I & 0\\ 0 & 0\end{bmatrix}\Lambda_1^{\,N(i)}\begin{bmatrix}I & 0\\ 0 & 0\end{bmatrix}\right)\mathbf{z}_0 \tag{4.15}$$

2. For the output-feedback NCS shown in Fig. 4-6, the augmented system equation is

$$\begin{bmatrix}\mathbf{x}(i_{k+1})\\ \widetilde{\mathbf{x}}(i_{k+1})\\ \overline{\mathbf{e}}(i_{k+1})\end{bmatrix}=\begin{bmatrix}\Phi & \Gamma L & -\Gamma L\\ L_oC & \Phi-L_oC+\Gamma L & -\Gamma L\\ L_oC & -L_oC & \Phi\end{bmatrix}\begin{bmatrix}\mathbf{x}(i_k)\\ \widetilde{\mathbf{x}}(i_k)\\ \overline{\mathbf{e}}(i_k)\end{bmatrix}. \tag{4.16}$$

Define $\Lambda_2=\begin{bmatrix}\Phi & \Gamma L & -\Gamma L\\ L_oC & \Phi-L_oC+\Gamma L & -\Gamma L\\ L_oC & -L_oC & \Phi\end{bmatrix}$, and $\mathbf{z}_2(i_k)=\begin{bmatrix}\mathbf{x}(i_k)\\ \widetilde{\mathbf{x}}(i_k)\\ \overline{\mathbf{e}}(i_k)\end{bmatrix}$, then we have

$$\mathbf{z}_2(i_{k+1})=\Lambda_2\mathbf{z}_2(i_k),\quad i_k\in[i_K,i_{K+1}). \tag{4.17}$$

The system described by (4.17) with the initial condition $\mathbf{z}_0=[\mathbf{x}(i_0)\quad \widetilde{\mathbf{x}}(i_0)\quad 0]^T$ has the following solution [27].

$$\mathbf{z}_2(i_k) = \Lambda_2{}^{i_k - i_K} \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N(i)} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \mathbf{z}_0 \tag{4.18}$$

The sufficient conditions for the stability of full-state-feedback and output-feedback NCSs are given as following two theorems.

Theorem 4-1: Let N_{max} denote the maximum value of $N(K)$. The sufficient condition under which the system (4.14) is globally exponentially stable around $\mathbf{z}_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ is that all the eigenvalues of $\left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1{}^{N_{\max}} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)$ are inside the unit circle.

Proof: Taking the norm of (4.15) we have

$$\left\| \mathbf{z}_1(i_k) \right\| = \left\| \Lambda_1{}^{i_k - i_K} \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1{}^{N(i)} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \mathbf{z}_0 \right\| \le \left\| \Lambda_1{}^{i_k - i_K} \right\| \cdot \left\| \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1{}^{N(i)} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \right\| \cdot \left\| \mathbf{z}_0 \right\| . \tag{4.19}$$

For the first term on the right-hand side, let $\bar{\sigma}(\Lambda_1)$ denotes the largest singular value of Λ_1. If $\bar{\sigma}(\Lambda_1) > 1$, we have

$$\left\| \Lambda_1{}^{i_k - i_K} \right\| \le \left(\bar{\sigma}(\Lambda_1) \right)^{i_k - i_K} \le \left(\bar{\sigma}(\Lambda_1) \right)^{N(K)} \le \left(\bar{\sigma}(\Lambda_1) \right)^{N_{max}} = K_1 . \tag{4.20}$$

If $\bar{\sigma}(\Lambda_1) \le 1$, we have $\left\| \Lambda_1{}^{i_k - i_K} \right\| \le \left(\bar{\sigma}(\Lambda_1) \right)^{i_k - i_K} \le 1$.

For the second term on the right-hand side of (4.19), we have

$$\left\| \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{N(i)} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \right\| \le \left\| \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{N_{max}} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \right\| = \left\| \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{N_{max}} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)^K \right\|,$$

which is bounded if all the eigenvalues of $\left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{N_{max}} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)$ are inside the unit circle. Thus

$$\left\| \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{N_{\max}} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)^K \right\| \le K_2 e^{-aK}, \tag{4.21}$$

with $K_2 > 0$ and $a > 0$.

Since there are time delays or packet losses, the event index K and the sampling time index i_k satisfy the following inequality.

$$\frac{i_k - 1}{N_{max}} \le K \le i_k. \tag{4.22}$$

Thus (4.21) can be rewritten as

$$\left\| \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{N_{max}} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right)^K \right\| \le K_2 e^{-ak} < K_2 e^{-a\frac{i_k - 1}{N_{max}}} = K_2 e^{\frac{a}{N_{max}}} e^{-\frac{a}{N_{max}} i_k} = K_3 e^{-b i_k}, \tag{4.23}$$

with $K_3 = K_2 e^{a/N_{max}} > 0$ and $b = a / N_{max} > 0$.

Thus from (4.19), (4.20) and (4.23) we can conclude

$$\left\|\mathbf{z}_1(i_k)\right\| = \left\|\Lambda_1^{\,i_k - i_K} \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \Lambda_1^{\,N(i)} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \mathbf{z}_0 \right\| \le K_1 \cdot K_3 e^{-bi_k} \cdot \left\|\mathbf{z}_0\right\|,$$

which implies the system (4.14) is globally exponentially stable around $\mathbf{z}_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ as $i_k \to \infty$.

Theorem 4-2: The system (4.17) is globally exponentially stable around $\mathbf{z}_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ if all the eigenvalues of $\left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2^{\,N_{max}} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)$ are inside the unit circle.

Proof: Taking the norm of (4.18) we have

$$\begin{aligned} \left\|\mathbf{z}_2(i_k)\right\| &= \left\|\Lambda_2^{\,i_k - i_K} \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2^{\,N(i)} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \mathbf{z}_0 \right\| \\ &\le \left\|\Lambda_2^{\,i_k - i_K}\right\| \cdot \left\| \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2^{\,N(i)} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \right\| \cdot \left\|\mathbf{z}_0\right\|. \end{aligned} \tag{4.24}$$

For the first term on the right-hand side, let $\overline{\sigma}(\Lambda_2)$ denotes the largest singular value of Λ_2. If $\overline{\sigma}(\Lambda_2) > 1$, we have

$$\left\|\Lambda_2^{\,i_k - i_K}\right\| \le \left(\overline{\sigma}(\Lambda_2)\right)^{i_k - i_K} \le \left(\overline{\sigma}(\Lambda_2)\right)^{N_{max}} = K_4. \tag{4.25}$$

If $\overline{\sigma}(\Lambda_2) \le 1$, we have $\left\|\Lambda_2^{\,i_k - i_K}\right\| \le \left(\overline{\sigma}(\Lambda_2)\right)^{i_k - i_K} \le 1$.

For the second term on the right-hand side of (4.24),

$$\left\| \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N(i)} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \right\| \le \left\| \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N_{max}} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \right\|$$

$$\le \left\| \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N_{max}} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)^{K} \right\|,$$

which is bounded if all the eigenvalues of $\left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N_{max}} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)$ are inside the unit circle. Thus

$$\left\| \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N_{max}} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)^{K} \right\| \le K_5 e^{-a_1 K}, \tag{4.26}$$

with $K_5 > 0$ and $a_1 > 0$. From (4-22) and (4-26), we have

$$\left\| \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N_{max}} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right)^{K} \right\| \le K_5 e^{-a_1 K} \le K_6 e^{-c i_k}, \tag{4.27}$$

with $K_6 = K_5 e^{a_1 / N_{max}} > 0$, and $c = a_1 / N_{max} > 0$.

Thus from (4.24), (4.25) and (4.27) we can conclude

$$\left\| z_1(i_k) \right\| = \left\| \Lambda_2{}^{i_k - i_K} \prod_{i=0}^{K} \left(\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \Lambda_2{}^{N(i)} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \mathbf{z}_0 \right\| \le K_4 \cdot K_6 e^{-c i_k} \left\| \mathbf{z}_0 \right\|,$$

which implies system (4.17) is globally exponentially stable around $\mathbf{z}_2 = [0 \quad 0 \quad 0]^T$ as $i_k \to \infty$.

Remark 4-2: In this section, the results given in [27] were extended to cover two-way-delay cases, and the upper bound of time delay was used instead of the delay itself to give the sufficient conditions of stability. These conditions can be verified more easily and practically, since the upper bound is a fixed value whereas the delay is usually randomly varying.

4.2.3 Experimental Verification

To verify the effectiveness of the algorithm developed in the previous section, several experiments by using the ball maglev test bed were conducted. The sporadic surges in time delays in the LANs between our labs were found to be bounded by 15 ms (see Fig. 4-2) which is 5 times as long as the sampling period, thus a 4-step-ahead estimator is designed, and the composition of a 56-byte-long IP packet transmitted from the server controller to the client actuator is shown in Fig. 4-7. It consists of a 20-byte-long IP header, an 8-byte-long UDP header, an 8-byte-long time stamp, one 4-byte-long current control signal, and four 4-byte-long estimated control signals.

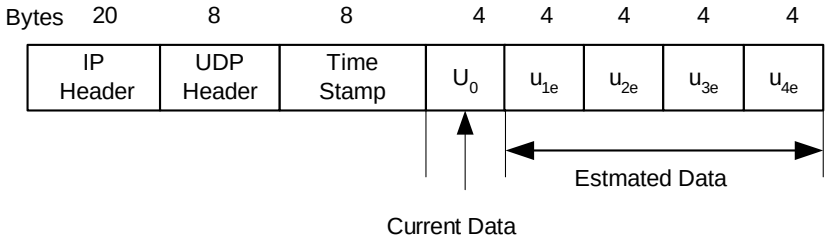


Fig. 4-7. Composition of a control-data packet from the controller to the actuator

In the first experiment, no compensation algorithm was used. At t = 10 s, we forced a data packet to be lost while transmitted from the server controller to the client actuator. With the introduction of this data-packet loss, no control signal will output to the actuator, it was expected that the system would become unstable. In the response of the system shown in Fig. 4-8, the 0 value after t = 10 s represents that the system indeed lost its stability. The ball could not maintain its equilibrium position and fell down.

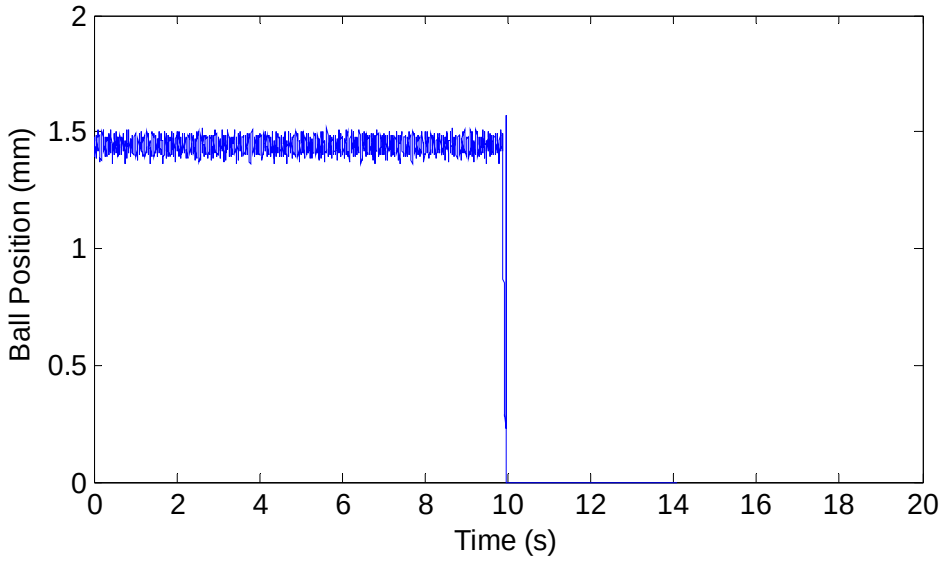


Fig. 4-8. Ball position with a packets loss occurring at t = 10 s without the compensation algorithm

In the second experiment, the estimator-based compensation algorithm was implemented, and one packet loss was introduced after every 4 successful data transmissions (i.e, at s 20% packet-loss rate) from t = 10 s onwards. As evident from the response shown in Fig. 4-9, the system remained stable with degraded performance and the ball did not fall down from its equilibrium position.

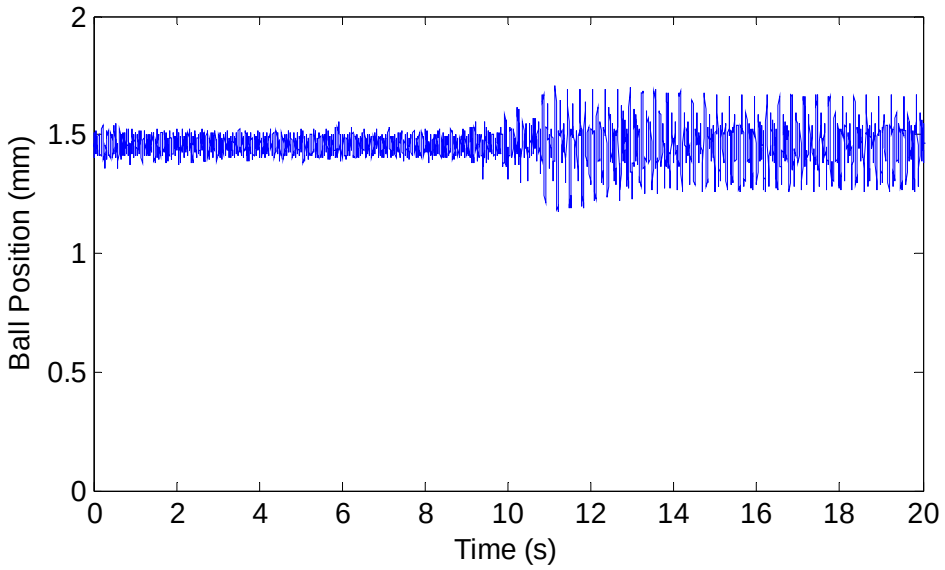


Fig. 4-9. Ball position with 20% packet losses occurring from t = 10 s onwards with the compensation algorithm

In Fig. 4-9, some hidden oscillations can be observed. In practical NCS applications, asynchronous and aperiodic sampling often occurs since the computer is time-shared or a part of a computer network, or there are technical imperfections in the instrumentation. However, asynchronous and aperiodic sampling is sometimes preferred over synchronous sampling when the former is applied intentionally to eliminate hidden oscillations.

In the third experiment, the estimator-based compensation algorithm was implemented and 4 consecutive packet losses were introduced once every 6 s from t = 10 s onwards. The response of the system is shown in Fig. 4-10. As evidenced for Fig. 4-10, the system maintained its stability successfully even in the event of 4 successive packet losses and the ball did not fall down from its equilibrium position. However, the periodic 1.1-mm peak-to-peak spikes in the ball position indicate the degraded system performance due to the packet losses and imperfect state estimation.

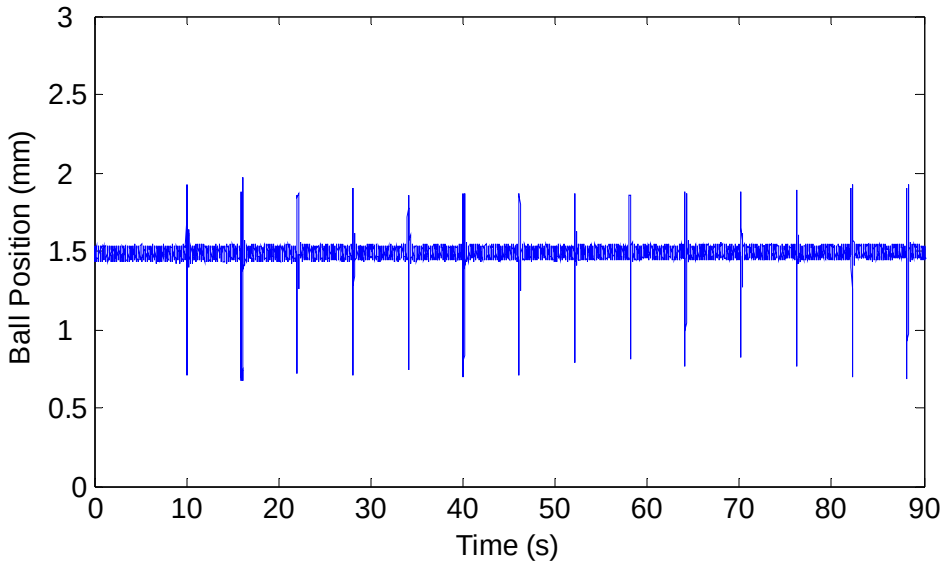


Fig. 4-10. Ball position with 4 successive packet losses occurring once every 6 s from t = 10 s onwards with the compensation algorithm

The above experimental results demonstrated that the system stability in the presence of time delays or packet losses could be maintained using the estimator-based compensation algorithm developed in the previous section.

4.3 Predictor- and Timeout-Scheme-Based Approach

This approach is based on a combination of a clock-driven sensor and actuator and event-driven controller. This is also a simple approach in which clock synchronization is not required. A multi-step-ahead control prediction algorithm based on an autoregressive (AR) model and timeout-scheme [21] is proposed to compensate for the time delays and packet losses in both the feedback path and the control-input path simultaneously.

4.3.1 Sensor Data Prediction

The plant and controller dynamics are modeled as

$$\begin{aligned} \mathbf{x}(i_{k+1}) &= \Phi\mathbf{x}(i_k) + \Gamma\mathbf{u}(i_k) + \mathbf{v}(i_k) \\ \mathbf{y}(i_k) &= C\mathbf{x}(i_k) + D\mathbf{u}(i_k) + \mathbf{w}(i_k) \end{aligned} \tag{4.28}$$

where $\mathbf{x} \in R^n$ is the state, $\mathbf{y} \in R^m$ is the output, and $\mathbf{u} \in R^v$ is the control input. A, B, C, and D are constant matrices of compatible dimensions.

To compensate for the delay τ_{sc}, a prediction algorithm is proposed for sensor data prediction in [22]. To predict the data, a procedure called system identification is used. There are two types of identification procedures. In off-line identification, a batch of data is collected from the system and this data is used to construct a model. Off-line identification is used for systems which have minimum changes in their behavior during operation.

For some other systems, it is necessary to design a model in order to support the decisions that have to be taken during their operation. Behaviors of these systems change considerably during operation. For these systems, data are updated continuously and it is necessary to infer the model at the same time as the data is collected. This identification procedure is known as recursive system identification [80]. Among various methods, an AR model and an autoregressive moving average (ARMA) model were used to predict the sensor data $\hat{\mathbf{y}}$ because of their simplicity [80]. Many experiments were conducted to select the best model to be used to predict the sensor data prediction. An AR model was finally chosen because the percentage error variation for this model is less than the percentage error variation for the ARMA model [21]. The AR model is defined as [22]

$$A(q^{-1})\mathbf{y}(k) = \hat{\mathbf{y}}(k+1), \tag{4.29}$$

where q^{-1} is the backward shift (or delay), $\mathbf{y}(k)$ corresponds to the k-th output, and $A(q^{-1})$ is defined as

$$A(q^{-1}) = a_1 q^{-1} + \ldots + a_n q^{-n} \quad . \tag{4.30}$$

Based on the recursive least-square methodology, an off-line identification of the parameters of the fifth-order AR model was performed using MATLAB [22].

It was also necessary to choose an appropriate order for the predictor. The accuracy of prediction and the number of computations required for prediction are the two important factors to consider in the decision of the order of predictors. It was observed that a tradeoff exists between the accuracy of a predictor and the number of computations done by the predictor for each prediction. Generally higher-order predictors have better accuracy of prediction than the lower-order ones of the same type. On the other hand, higher-order predictors take more number of computations for a prediction than the lower-order predictors of the same type. A good predictor should give a reasonably accurate prediction with the minimum usage of computational resources.

Adequate amount of off-line output data of the stable system was required for development of predictors. To collect on-line sensor data of the stable system, the test bed was operated with the feedback loop closed locally. The sensor data of the system for 17162 samples were collected. MATLAB used half of these data to develop predictors and the rest half for their validation. The percentage of the sensor variations reproduced by the predictors is known as fit [80]. Mathematically, a fit can be stated as

$$fit = \left[\frac{1 - norm(Y - \hat{Y})}{norm(Y - mean(Y))} \right] \cdot 100 \tag{4.31}$$

where Y is the measured output and $\hat{Y}$ is the predicted model output. A higher best fit implies better prediction. Table 4-2 represents the best-fit values for auto-regressive (AR) models for various step-ahead predictors [22].

Table 4-2. Best fits for AR models [22]

Order	1-step	2-step	3-step	4-step	5-step	6-step	7-step	8-step
8	74.43	67.02	65.40	65.60	64.78	63.98	64.86	64.39
7	72.63	64.79	63.76	63.98	62.99	60.96	60.36	60.04
6	72.05	62.68	61.35	62.16	61.27	59.67	57.34	55.92
5	71.95	62.86	61.74	62.05	60.43	59.61	57.64	56.97

Based on the above discussion and simulation results, an 8th-order 4-step-ahead predictor was designed after numerous design iterations. The 4-step-ahead prediction for the control signal was calculated using the predicted sensor data. The parameter vectors for various predictors, calculated using MATLAB were used to develop the following prediction equations for our maglev test bed [22].

$$\begin{aligned}
&\hat{y}(t+1) = 0.8122y(t) - 0.3479y(t-1) - 0.0294y(t-2) + 0.4605y(t-3) + 0.0742y(t-4) \\
&+ 0.1042y(t-5) + 0.1117y(t-6) - 0.3561y(t-7) \\
&\hat{y}(t+2) = 0.3117y(t) - 0.3119y(t-1) + 0.4366y(t-2) + 0.44482y(t-3) + 0.1645y(t-4) \\
&+ 0.1964y(t-5) - 0.2653y(t-6) - 0.2892y(t-7) \\
&\hat{y}(t+3) = -0.0587y(t) + 0.3281y(t-1) + 0.4390y(t-2) + 0.3080y(t-3) + 0.2195y(t-4) \\
&- 0.2329y(t-5) - 0.2544y(t-6) - 0.1110y(t-7) \\
&\hat{y}(t+4) = 0.2804y(t) + 0.4594y(t-1) + 0.3097y(t-2) + 0.1925y(t-3) - 0.2372y(t-4) \\
&- 0.2605y(t-5) - 0.1176y(t-6) + 0.0209y(t-7)
\end{aligned} \tag{4-32}$$

4.3.2 Timeout Scheme

To compensate for the time-delay τ_{ca}, a time-out scheme was developed. In this timeout scheme, the event-driven controller computes the control input and predicts the 4-step-ahead

control data as soon as the new sensor data are available at the controller node. The controller sends the current control data together with the predicted control data as a package to the actuator node. At the actuator node, if no new control package is available within the t_0 timeout threshold, the actuator calls a timeout and the predicted control data stored in the previous control package is used as control input. For the open-loop unstable ball maglev test bed, it was calculated that if the sampling period is 3 ms, the actuation has to occur within 1.42 ms after sampling for the system stability [21]. Thus we choose 1.42 ms as the timeout threshold.

Figs. 4-11 and 4-12 show two examples of implementing the time-out scheme to compensate for two-way time delays and data-packet losses. In these two figures, the label y denotes the sensor data transmitted from the client sensor to the server controller, the label u denotes the control signal data transmitted from the server controller to the client actuator, and the label t_0 denotes the timeout threshold.

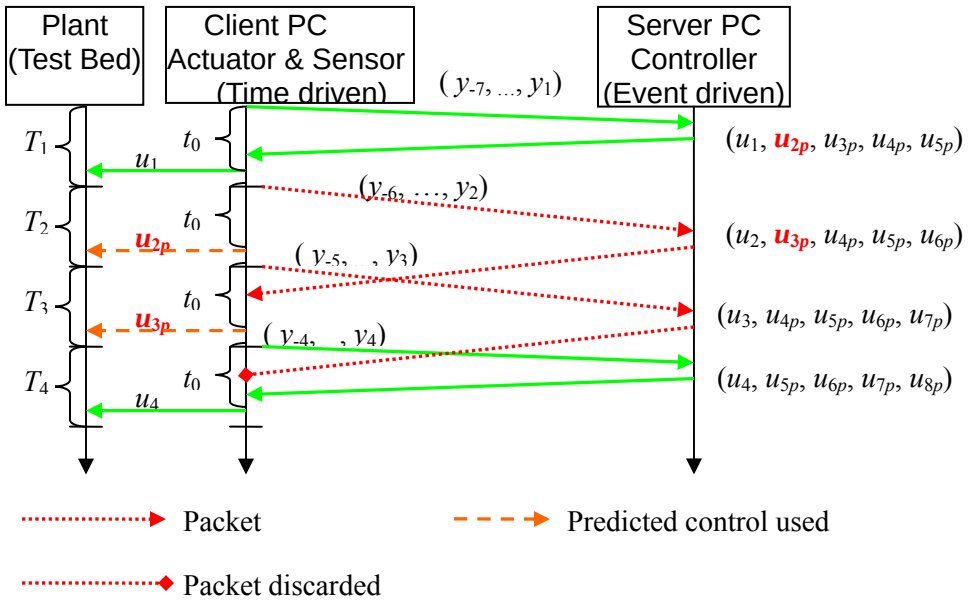


Fig. 4-11. Communication process with two-way time delays

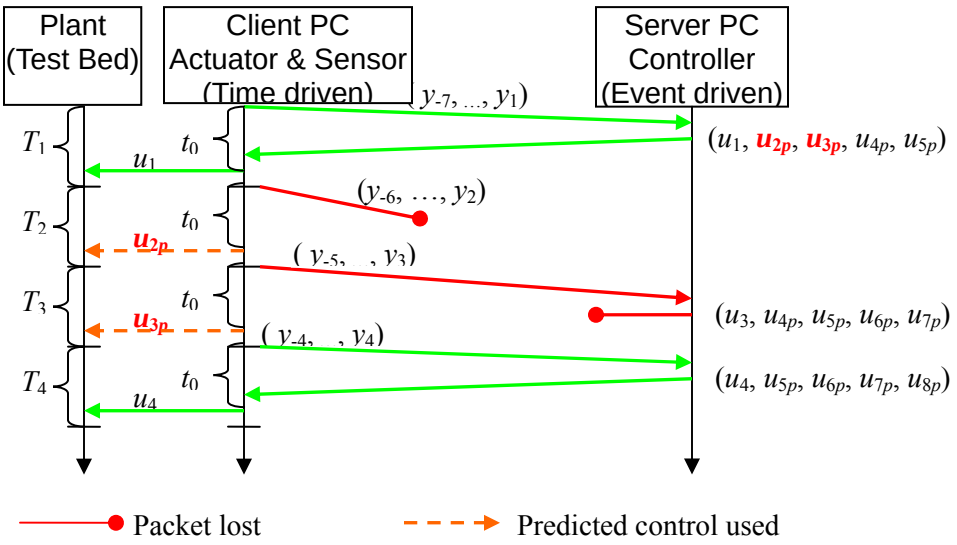


Fig. 4-12. Communication process with two-way packet losses

In Figs. 4-11 and 4-12, all data-packets are time stamped, the subscripts of the labels denote the sampling-period indices and indicate whether the data are predicted ($_p$). For example, y_2 is the sensor data of the second sampling period, u_3 is the control data for the third sampling period, and u_{4p} is the predicted control data for the fourth sampling period. The solid arrows originating from the client side towards the server side denote the transmission of the sensor-data packets that reach the server side. The solid arrows starting from the server side towards the client side denote the control-data packet inputs. The round-tip arrows indicate lost data communication in a given sampling period, and the dotted arrows, delayed communication. The dashed arrows indicate that the formerly predicted control input is applied when the actual current control-data packet does not reach the client side before the timeout is called. The square-tip arrows indicate the delayed control-data packet of the previous sampling period is discarded if there is a new control-data packet available before the timeout is called. Thus most recent control data such as u_4 shown in Fig. 4-11 are used.

Thus the proposed compensation algorithm can dealt with time delays and packet losses in both the feedback path and control forward path simultaneously. In case of out-of-order transmission of packets, the outdated packets are simply discarded as in the estimator-based algorithm proposed in Section 4.2. For instance, if the (n + 1)-th control-data packet arrives earlier than the n-th packet at the actuator node from the controller node, the actuator node neglects the n-th packet and uses the most recent (n + 1)-th control-data packet. By this, this algorithm can deal with out-of-order packet transmission as well. Thus a control input to the plant is always generated in each sampling period with either actual or predicted control data depending on the actual data's availability.

4.3.3 UDP Packet Composition

The composition of a typical 68-byte-long sensor-data packet transmitted from the client to the server is shown in Fig. 4-13. A time stamp is taken on the client side at sampling and is sent to the server. The server does not modify the time stamp but sends it back to the client along with the calculated control data. This time stamp is then used by the client to identify whether the arrived data packet is the expected packet or a delayed packet. Out-of-order data-packet is simply discarded in this compensation scheme.

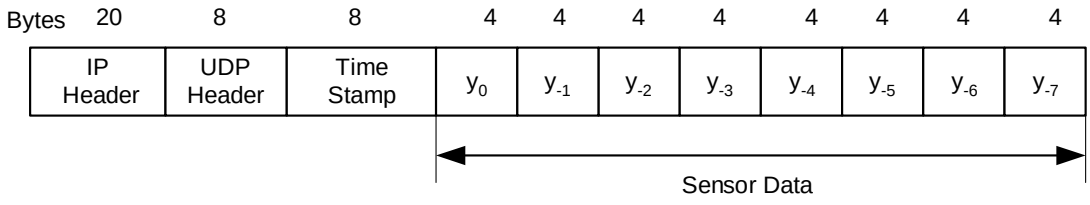


Fig. 4-13. Composition of a sensor-data-packet transmitted from the sensor to the controller

The composition of a typical 56-byte-long control-data packet transmitted from the server to the client is shown in Fig. 4-14. It consists of a 20-byte-long IP header, an 8-byte-long UDP header, an 8-byte-long time stamp, one 4-byte-long current control-data value, and four 4-byte-long predicted control-data values.

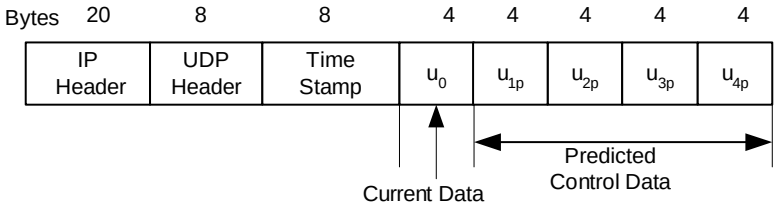


Fig. 4-14. Composition of a control-data-packet transmitted from the server controller to the client actuator

4.3.4 Experimental Verification

Three experiments similar to those in Section 4.2.2 were conducted. The experiment setup and condition are exact the same as those in Section 4.2.2. The first experiment is exact the same with the first experiment in Section 4.2.2, no compensation algorithm was used. At t = 12 s, we forced the sensor-data packet to be lost while transferred from the client to the server, then a zero control input was applied to the actuator. With the introduction of this packet loss, it was expected that the system would become unstable as in the first experiment in Section 4.2.2. The response of the system is shown in Fig. 4-15 which is very similar to the Fig. 4-8 as expected. In Fig. 4-15, the 0 value of the vertical axis indicates that the ball could not maintain its equilibrium position and fell down. This happened because of the introduced packet loss. The system could not remain stable in the absence of the compensation.

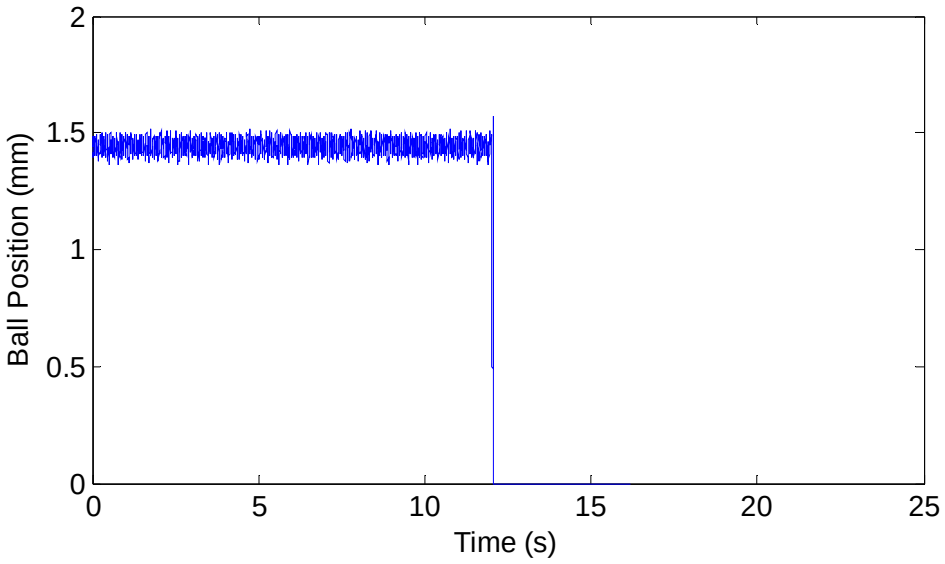


Fig. 4-15. Ball position profile with packet loss occurring at t = 12 s without compensation

In the second experiment, the predictor-based compensation algorithm was implemented. From t = 12 s onwards, artificial packet loss was introduced at every fifth sample (i.e. 20% packet loss rate). The response of the system is shown in Fig. 4-16. As evidenced from Fig. 4-16, the system remained stable throughout the experiment and the ball did not fall down from its equilibrium position. However, the increased movement of the ball about the equilibrium point indicates the performance degrade due to the packet losses and the multi-step-ahead sensor- and control-data prediction that would inevitably introduce prediction error.

In the third experiment, the predictor-based compensation algorithm was used and 4 successive sensor-data-packet losses occurred once every 12 s from t = 2 s onwards. The response of the system is shown in Fig. 4-17. The system remained stable throughout the experiment even in the event of 4 successive packet losses, and the ball did not fall down from its equilibrium position. This experimental result demonstrates that this algorithm is effective to

maintain the system stability with up to 4 successive packet losses or time delays as long as 4 sampling periods.

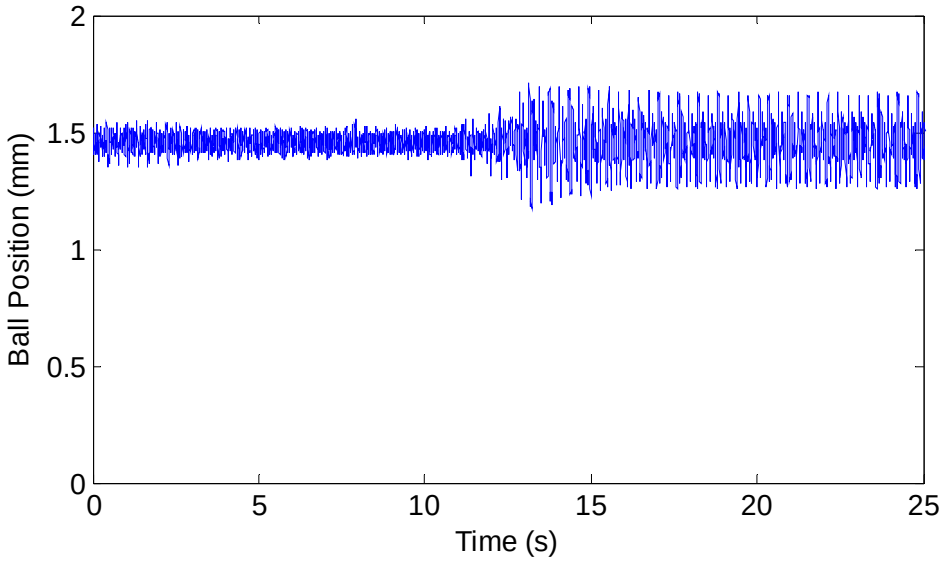


Fig. 4-16. Ball position profile with average 20% packet loss occurring after t = 12 s

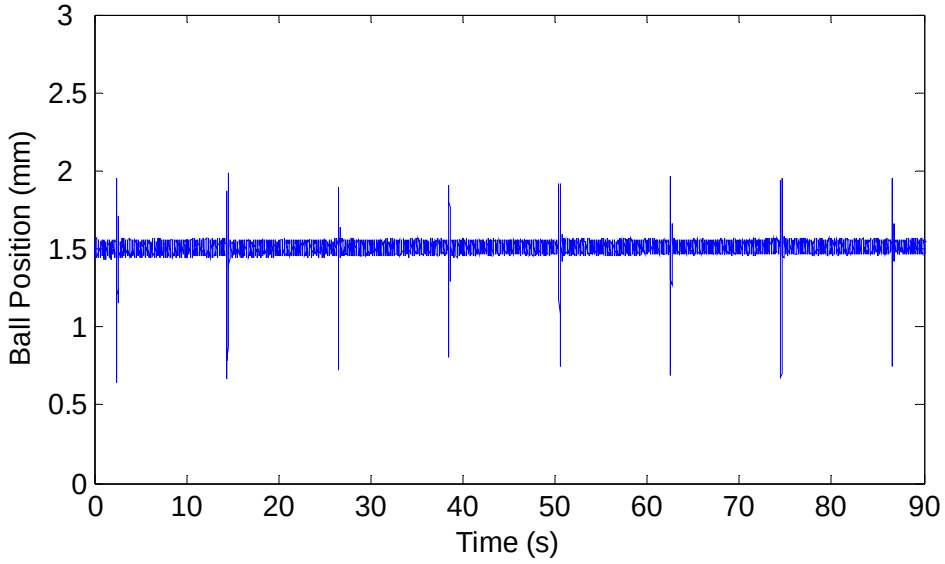


Fig. 4-17. Ball position profile with 4 successive packet losses occurring every 12 s after t = 2 s

The above experimental results verify that the prediction- and timeout-scheme-based compensation methodology developed in Section 4.3.2 is also effective to maintain the stability of network-based real-time control systems in the event of data-packet losses.

By comparing Fig. 4-9 with Fig. 4-16, and Fig. 4-10 with Fig. 4-17, it is concluded that the performance of these two compensation algorithms proposed in this chapter is very close in terms of maintaining system stability. To further investigate their performances, additional experiments were conducted in the following section.

4.4 Experiments for System QoC Verification

With the ball maglev system as a real-time NCS test bed and the time-delay/packet-loss compensation algorithms proposed in previous sections, we successfully implemented a real-time control over the Ethernet. Several additional experiments were conducted to determine the performance of these two algorithms and the degradation of system performance due to time delays or packet losses in step response and dynamic tracking. Two compensation algorithms were both verified. The first set of experiments was step response. Fig. 4-18 (a) shows a closed-loop step (started at t = 7 s) response of the test setup without packet loss. Fig. 4-18 (b) shows the step (started at t = 7 s) response of the system implemented with model-estimation-based algorithm for packet loss compensation. Fig. 4-18 (c) shows the step (started at t = 7 s) response of the system implemented with sensor-data-prediction and timeout-scheme-based algorithm for packet-loss compensation. The average packet-loss rate is 20%. The closed-loop systems with packet losses were stable with a worse stability margin due to packet losses and imperfect state and control estimation or prediction.

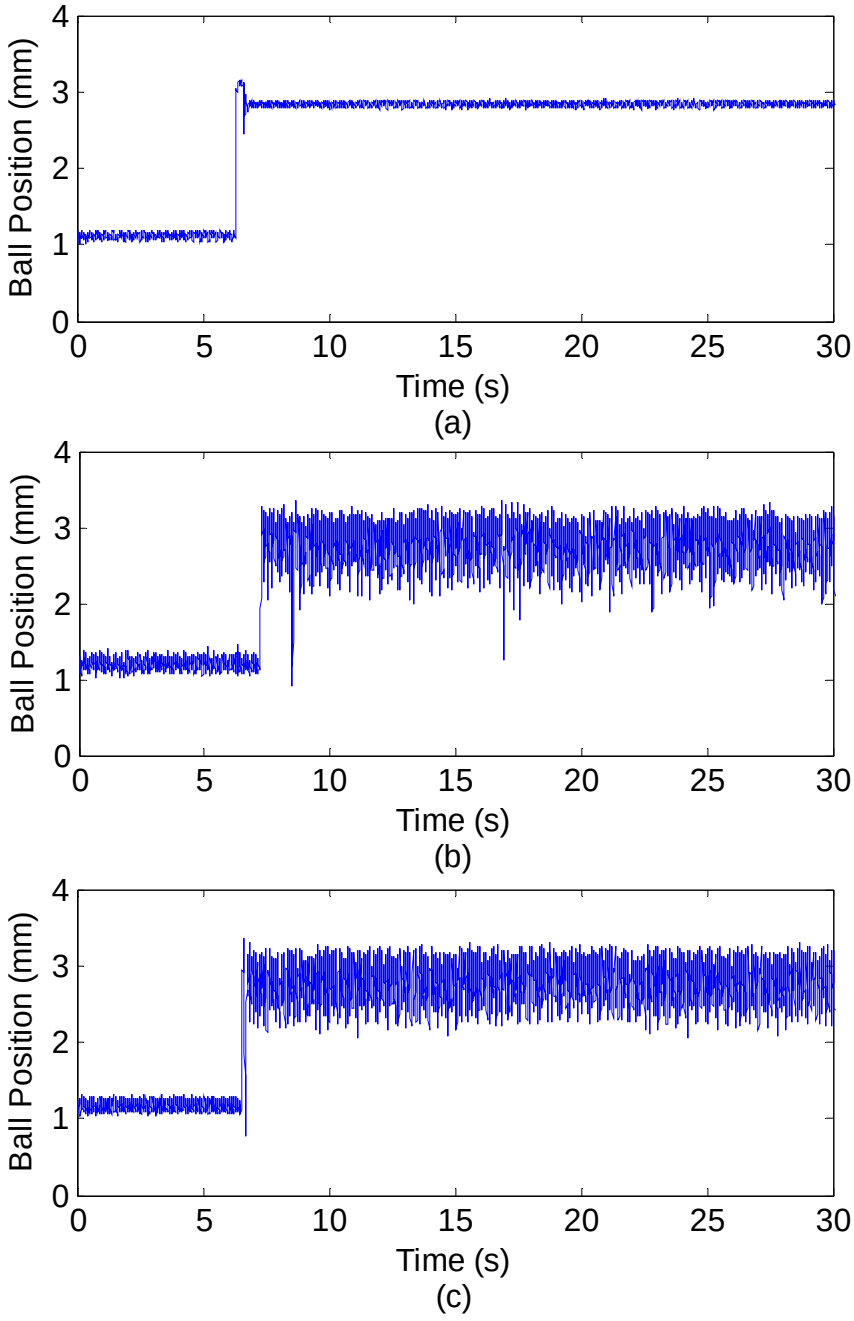


Fig. 4-18. Step responses of the ball maglev system (a) without packet loss, (b) with the model-estimation-based algorithm and 20% packet losses, and (c) with the prediction- and timeout-scheme-based algorithm and 20% packet losses

The second set of experiments was to make the ball track commanded trajectories. Fig. 4-19 (a) shows the system responses of tracking a sinusoidal position command at the average packet-loss rate of 20% from t = 50 s onwards in the system with model-estimation-based compensation algorithm. Fig. 4-19 (b) shows the system response of tracking a sinusoidal position command without packet loss in the system with prediction and timeout scheme-based compensation algorithm. Fig. 4-19 (c) shows the system response of tracking a sinusoidal position command at the loss average packet-loss rate of 20% in the system with prediction and timeout scheme-based compensation algorithm.

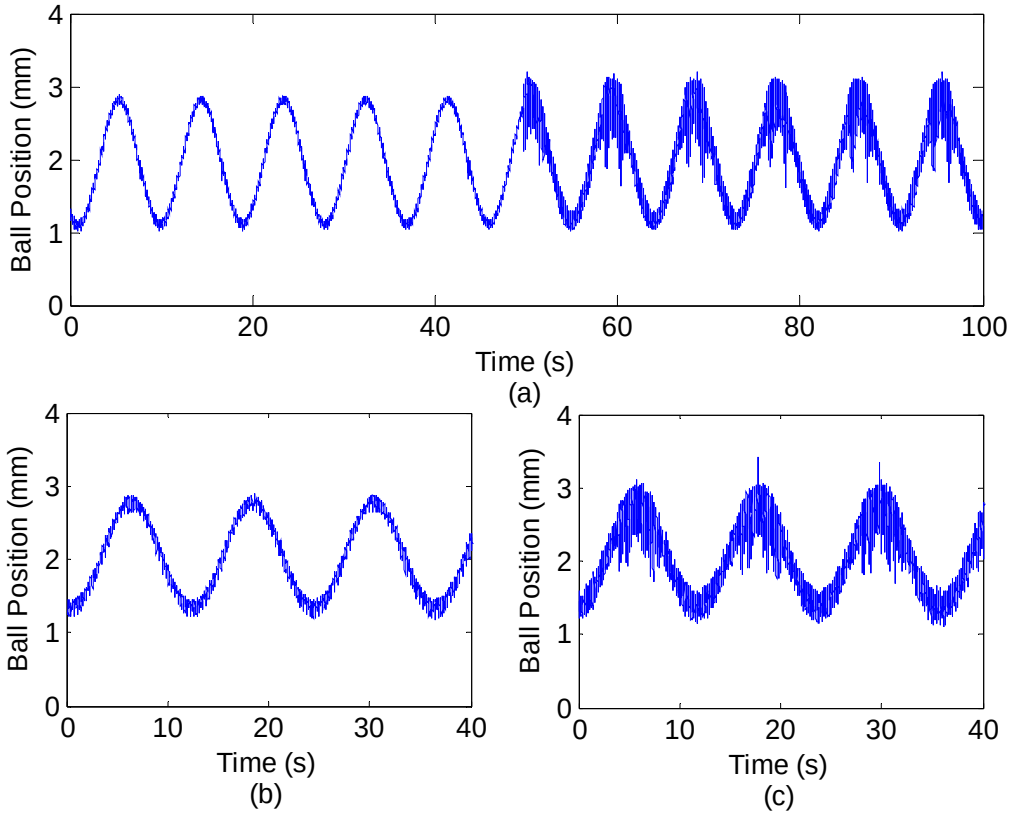


Fig. 4-19. The ball maglev system responses of tracking a sinusoidal command (a) with 20% packet losses beginning at t = 50 s with model-estimation-based algorithm, (b) without packet loss, and (c) with 20% packet losses with the prediction- and timeout-scheme-based algorithm

Fig. 4-20 shows the ball maglev system responses of tracking a saw-tooth position command at different packet-loss rates from t = 34 s onwards with the model-estimation-based algorithm. From 4-20, it can be shown that the fluctuation in the ball position increases with the increase of the packet-loss rate.

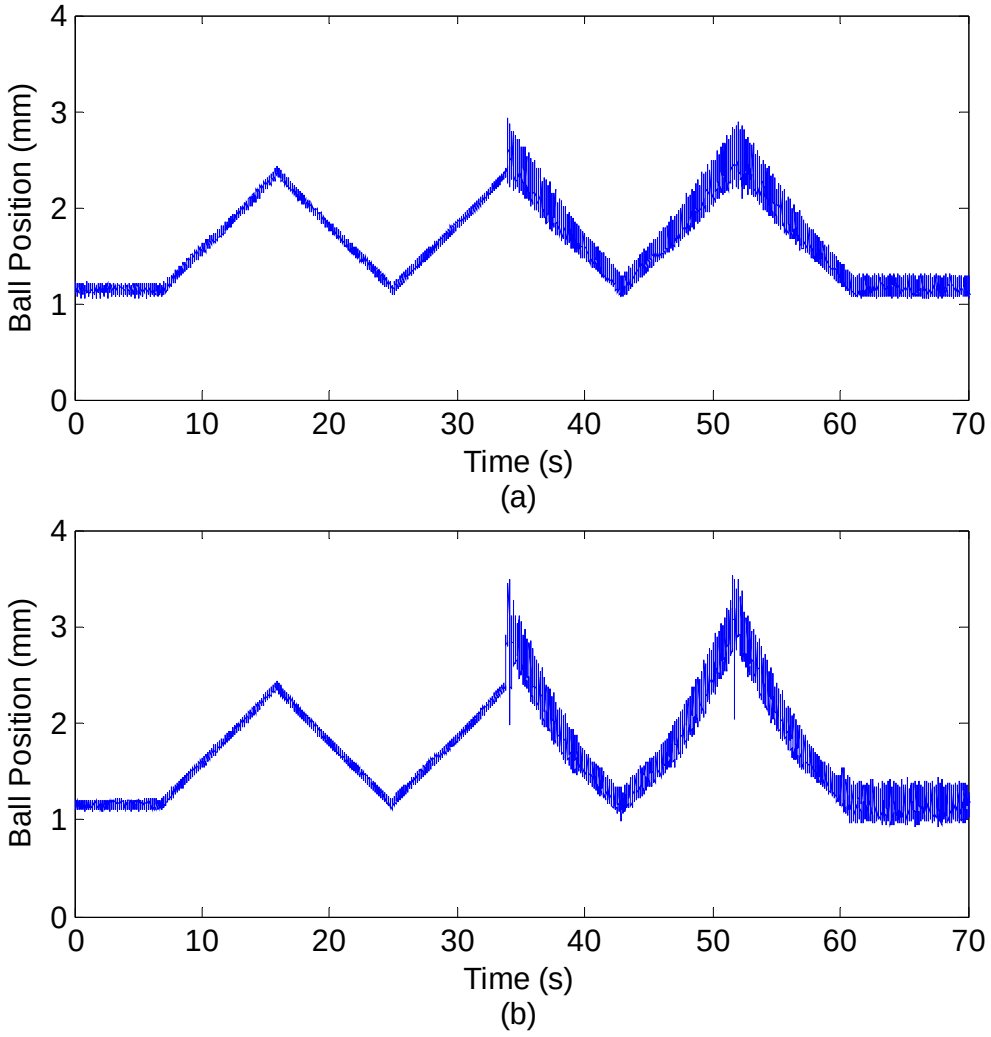


Fig. 4-20. The ball maglev system response of tracking a saw-tooth position command with (a) 10% packet losses (b) 20% packet losses

4.5 Summary

This chapter presents solutions to time-delay and packet-loss problems encountered in real-time operation of an open-loop unstable ball maglev test bed via an Ethernet LAN. A novel model-estimation based algorithm and a sensor-data prediction with timeout-scheme based algorithm were developed to compensate for the two classes of time delays and packet losses simultaneously. Experiment results verified the feasibility and effectiveness of these two algorithms.

For model-estimation based algorithm, the parameter p depends on the characteristics of the NCS and the accuracy of the plant model for state estimation. Assuming a larger p can maintain system stability in the present of longer time delays. However it will lead to excessive computation time and the size of control signal package will increase. This will use up more of the communication bandwidth and cause longer time delay. Thus there is an engineering trade-off between the value of p and the overall NCS performance. Augmented system model with the model-estimation based algorithm was analyzed and the system stability of the compensated system was presented.

For sensor-data prediction based algorithm, there also exists a trade-off between the accuracy of a predictor and the number of computations done by the predictor for each prediction. Generally higher-order predictors have better accuracy of prediction than the lower-order ones of the same type. On the other hand, higher-order predictors take more number of computations for a prediction than the lower-order predictors of the same type. A good predictor should give a reasonably accurate prediction with the minimum usage of computational resources.

CHAPTER V

ADVANCED CONTROL DESIGN FOR NCS

After successfully implemented a real-time control of a ball maglev test bed via the Ethernet in the previous chapter, we propose advanced control design methodologies in this chapter to further improve the QoC of NCSs.

5.1 Introduction

Unlike conventional control systems, an NCS essentially comprises multiple nodes communicating with each other over communication networks. The successful design and implementation of an NCS requires an appropriate integration of control systems, real-time systems, and network communication systems through co-design. For advanced control design, modern control theories like robust control and optimal control have been applied to NCSs to further improve the control performance.

5.2 Robust Control of NCSs with Uncertainties

In this section, robust H_∞-control problems for NCSs with network-induced time delays and subject to norm-bounded parameter uncertainties are presented and solved. Based on a new discrete-time model, two approaches of robust controller design are proposed—design of a memoryless state-feedback controller and design of a dynamic state-feedback controller. The

proposed memoryless state-feedback controller design method is given in terms of LMIs [41], and the delay bound can be computed by using the standard LMI techniques. A numerical example is given to illustrate the feasibility and effectiveness of this methodology. The proposed dynamic state-feedback controller design method is based on a discrete-time Artstein transform [31]. With the sufficient conditions for robust stability and H_∞ control developed in this dissertation, the upper bound of the network-induced time delays that can be used as a guideline in choosing proper networks as communication media for the NCS design are also derived.

Since typical NCSs are discrete-time systems, it is more natural to analyse NCSs with discrete-time models. In this paper, we consider an NCS framework as shown in Fig. 5-1. We assume that the time for AD conversion at the sensor node and DA conversion at the actuator node is negligible compared to the communication delays.

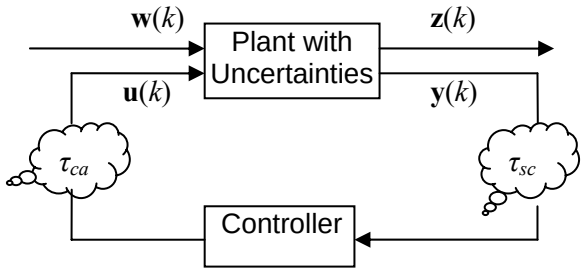


Fig. 5-1. Networked control system with network-induced time delays

Consider the NCS shown in Fig. 5-1 with parameter uncertainties and time delays described by the following discrete-time state equations.

$$\begin{aligned} \mathbf{x}(k+1) &= (A + A_u)\mathbf{x}(k) + (B + B_u)\mathbf{u}(k - h_k) + B_w\mathbf{w}(k) \\ \mathbf{x}(m) &= \boldsymbol{\theta}(m), \forall m \in \begin{bmatrix} -h_0 & 0 \end{bmatrix} \\ \mathbf{z}(k) &= E\mathbf{x}(k), \end{aligned} \tag{5.1}$$

where $\mathbf{x}(k) \in R^n$ is the state, $\mathbf{u}(k) \in R^m$ is the control input, $\mathbf{w}(k) \in R^p$ is the disturbance input, h_k is an integer that denotes the number of the sampling periods as the length of time delay at time instant k. $\boldsymbol{\theta}(\cdot)$ is the initial condition, $\mathbf{z}(k)$ is the controlled output, A, B, B_w, and E are known real constant matrices of appropriate dimensions that describe the nominal system, and A_u and B_u are real matrix functions representing time-varying parameter uncertainties. The network sampling period is h.

In this chapter, we make the following assumptions:

5-1. The admissible uncertainties are assumed to be of the form

$$A_u = L_A F_A(k) E_A, \quad B_u = L_B F_B(k) E_B. \tag{5.2}$$

We define $\overline{A}(k) = A + L_A F_A(k) E_A$ and $\overline{B}(k) = B + L_B F_B(k) E_B$, where $F_A(k)$ and $F_B(k)$ are unknown real time-varying matrices with Lebesgue measurable elements satisfying

$$\|F_A(k)\| \le 1; \quad \|F_B(k)\| \le 1, \ \forall k, \tag{5.3}$$

and L_A, L_B, E_A, and E_B are known real constant matrices that characterize how the uncertain parameters in $F_A(k)$ and $F_B(k)$ enter the nominal matrices A and B.

5-2. The upper bound of the network-induced time delays is H sampling periods.

5-3. The sampling frequency is high enough, and there exists a real scalar $S > 0$ such that for any real symmetric positive-definite matrix P, the following inequality holds.

$$S + \mathbf{x}(k)^T P \mathbf{x}(k) - \left[\mathbf{x}(k) - \mathbf{x}(k+1)\right]^T P \left[\mathbf{x}(k) - \mathbf{x}(k+1)\right] > 0 \tag{5.4}$$

The following robust control problems are addressed in this chapter.

1. Robust Stabilization: Given $H > 0$, find a linear state-feedback control law for the system (5.1–5.3) such that the resulting closed-loop system is robustly stable for any time delay length parameter h_k satisfying $h_k \le H$ for all k.
2. Robust H_∞ Control: Given scalars $\gamma > 0$ and $H > 0$, find a linear state-feedback control law for the system (5.1–5.3) such that the resulting closed-loop system is robustly stable with disturbance attenuation γ for any time delay length parameter h_k satisfying $h_k \le H$ for all k. That is, the closed-loop system satisfies the following inequality

$$\left\| T_{zw} \right\|_\infty \le \gamma \ , \tag{5.5}$$

   where $\left\| T_{zw} \right\|_\infty$ is the H_∞ norm of the transfer function T_{zw} from the disturbance input $\mathbf{w}(k)$ to the controlled output $\mathbf{z}(k)$.
3. Robust Parameter Optimization: Based on the results of the above robust-control problems 1 and 2, there are two robust parameter-optimization problems: (a) given γ, find the largest H, i.e. determine an upper bound for the time delay such that the uncertain system (5.1–5.3) is robustly stabilizable with a prescribed level of disturbance attenuation for any time delay no longer than this bound and for admissible uncertainties, and (b) given H, find the smallest γ, i.e. determine the lower bound of disturbance attenuation for the uncertain system (5.1–5.3).

The objective is to develop an admissible controller for robust-control problems 1, 2, and 3. Suitable state-feedback controllers are developed through two different approaches, i.e. memoryless and dynamic state-feedback control.

The following Lemma is used in this chapter.

Lemma 5-1: Let *A*, *L*, *E*, and *F* be real matrices of appropriate dimensions with $\|F\| \le 1$. Then the following inequalities hold.

(1) For any real symmetric positive-definite matrix *P* and scalar $s > 0$ such that $sI - EPE^T > 0$,

$$(A + LFE)P(A + LFE)^T \le APA^T + APE^T(sI - EPE^T)^{-1}EPA^T + sLL^T.$$

(2) For any real symmetric positive-definite matrix *P* and scalar $s > 0$ such that $P - sLL^T > 0$,

$$(A + LFE)^T P^{-1}(A + LFE) \le A^T(P - sLL^T)^{-1}A + s^{-1}E^T E.$$

(3) For any scalar $s > 0$,

$$LFE + E^T F^T L^T \le s^{-1}LL^T + sE^T E$$

Proof: To prove Part (1) introduce the matrix

$$N = (s^{-1}I - EPE^T)^{-\frac{1}{2}}EPA^T - (s^{-1}I - EPE^T)^{\frac{1}{2}}F^T L^T.$$

The expansion of $N^T N \ge 0$ leads to

$$APE^T F^T L^T + LFEPA^T + LFEPE^T F^T L^T \le APE^T(s^{-1}I - EPE^T)^{-1}EPA^T + s^{-1}LFF^T L^T.$$

By considering the fact that $F^T F \le I$ and using the matrix inversion lemma, the desired result in Part (1) follows immediately. Part (2) can be easily proved by the similar argument as in the proof of Part (1). Part (3) is a well known fact [81].

For controller design, first we propose a memoryless controller to solve robust-control problems 1, 2, and 3 for system (5.1–5.3),

$$\mathbf{u}(k) = K\mathbf{x}(k), \tag{5.6}$$

where $K \in R^{m\times n}$ is a constant matrix.

5.2.1 Robust Stabilization

Assuming $\mathbf{w}(k) = 0,$ we present a sufficient condition as following two theorems for the existence of memoryless state-feedback controllers for the uncertain system (5.1–5.3).

Theorem 5-1: Given $H > 0$, the system (5.1–5.3) is robustly closed-loop stable with a control input of (5.6) for any time delay length parameter h_k satisfying $h_k \le H$ if there exist real symmetric positive-definite matrices P, P_1, P_2, and Q, a matrix K, and scalars $r_i > 0$, $i = 1, \ldots, 5$, such that following inequalities hold

$$R(P, P_1, P_2, Q, K, H, r_i) < 0 \tag{5.7}$$

$$r_3 I - E_B Q E_B{}^T > 0, \;\; P_1 - r_4 L_A L_A{}^T > 0, \;\; P_2 - r_5 L_B L_B{}^T > 0 \tag{5.8}$$

$$P^{-1} - P_1 - P_2 \ge 0, \;\; Q - K P^{-1} K^T \ge 0, \tag{5.9}$$

where

$$\begin{aligned} & R(P,\ P_1,\ P_2,\ Q,\ K,\ H,\ r_i) \\ & \equiv (A+BK)^T P + P(A+BK) + P(r_1 L_A L_A{}^T + r_2 L_B L_B{}^T)P + r_1^{-1} E_A{}^T E_A + r_2^{-1} K^T E_B{}^T E_B K \\ & + HP\left[BQB^T + r_3 L_B L_B{}^T + BQE_B{}^T (r_3 I - E_B Q E_B{}^T)^{-1} E_B Q B^T\right] P \\ & + H\left[(A-I)^T (P_1 - r_4 L_A L_A{}^T)^{-1}(A-I) + r_4^{-1} E_A{}^T E_A + K^T B^T (P_2 - r_5 L_B L_B{}^T)^{-1} BK + r_5^{-1} K^T E_B{}^T E_B K\right]. \end{aligned} \tag{5.10}$$

Proof: From (5.1–5.2) and (5.6), we obtain $\mathbf{x}(k+1) - \mathbf{x}(k) = (\overline{A}(k) - I)\mathbf{x}(k) + \overline{B}(k) K \mathbf{x}(k - h_k)$, and

$$\mathbf{x}(k-h_k)=\mathbf{x}(k)-\sum_{i=-h_k}^{-1}\left[\mathbf{x}(k+i+1)-\mathbf{x}(k+i)\right]=\mathbf{x}(k)-\sum_{i=-h_k}^{-1}\left[(\overline{A}(k+i)-I)\mathbf{x}(k+i)+\overline{B}(k+i)K\mathbf{x}(k-h_k+i)\right].$$

Then,

$$\mathbf{x}(k+1)-\mathbf{x}(k)=(\overline{A}(k)+\overline{B}(k)K-I)\mathbf{x}(k)-\overline{B}(k)K\sum_{i=-h_k}^{-1}\left[(\overline{A}(k+i)-I)\mathbf{x}(k+i)+\overline{B}(k+i)K\mathbf{x}(k-h_k+i)\right].$$

(5.11)

Define a discrete-time Lypunov function as

$$V(\mathbf{x},k)=\mathbf{x}^T(k)P\mathbf{x}(k)+W(\mathbf{x},k)+S(\mathbf{x},k), \tag{5.12}$$

where P is a real symmetric positive-definite matrix, and

$$\begin{aligned}W(\mathbf{x},k)\equiv&\sum_{j=-h_k}^{-1}\sum_{i=j}^{-1}\mathbf{x}^T(k+i)\left[\overline{A}(k+i)-I\right]^T P_1^{-1}\left[\overline{A}(k+i)-I\right]\mathbf{x}(k+i)\\&+\sum_{j=-h_k}^{-1}\sum_{i=j-h_k}^{-1}\mathbf{x}^T(k+i)\left[K^T\overline{B}^T(k+i+h_k)P_2^{-1}\overline{B}(k+i+h_k)K\right]\mathbf{x}(k+i),\end{aligned} \tag{5.13}$$

$$S(\mathbf{x},k)\equiv S(\mathbf{x},0)+\sum_{i=-k}^{-1}\mathbf{x}^T(k+i)P\mathbf{x}(k+i)-\sum_{i=-k}^{-1}\left[\mathbf{x}(k+i)-\mathbf{x}(k+i+1)\right]^T P\left[\mathbf{x}(k+i)-\mathbf{x}(k+i+1)\right], \tag{5.14}$$

where P_1 and P_2 are real symmetric positive-definite matrices to be chosen. Based on Assumption 5-3, we have $W(\mathbf{x}, k) > 0$ and $S(\mathbf{x}, k) > 0$. From (5.12), we obtain

$$\Delta V(\mathbf{x},k)=V(\mathbf{x},k+1)-V(\mathbf{x},k)=\Delta X+\Delta W+\Delta S, \tag{5.15}$$

where

$$\begin{aligned}\Delta X &= \mathbf{x}^T(k+1)P\mathbf{x}(k+1) - \mathbf{x}^T(k)P\mathbf{x}(k),\\ \Delta W &= W(\mathbf{x},k+1) - W(\mathbf{x},k),\\ \Delta S &= S(\mathbf{x},k+1) - S(\mathbf{x},k).\end{aligned} \tag{5.16}$$

For ΔX, we have

$$\begin{aligned}\Delta X &= \mathbf{x}^T(k)P\left[\mathbf{x}(k+1)-\mathbf{x}(k)\right] + \left[\mathbf{x}(k+1)-\mathbf{x}(k)\right]^T P\mathbf{x}(k) + \left[\mathbf{x}(k+1)-\mathbf{x}(k)\right]^T P\left[\mathbf{x}(k+1)-\mathbf{x}(k)\right]\\ &= \mathbf{x}^T(k)P\left[\mathbf{x}(k+1)-\mathbf{x}(k)\right] + \left[\mathbf{x}(k+1)-\mathbf{x}(k)\right]^T P\mathbf{x}(k) + \Delta X_1,\end{aligned} \tag{5.17}$$

where $\Delta X_1 = \left[\mathbf{x}(k+1)-\mathbf{x}(k)\right]^T P\left[\mathbf{x}(k+1)-\mathbf{x}(k)\right]$. Substituting (5.11) into (5.17), and applying Lemmas 3-1 and 5-1, we obtain

$$\begin{aligned}&\Delta X = \Delta X_1 + \left[(\overline{A}(k)+\overline{B}(k)K-I)\mathbf{x}(k)\right]^T P\mathbf{x}(k)\\ &-\left[\overline{B}K\sum_{i=-h_k}^{-1}\left[(\overline{A}(k+i)-I)\mathbf{x}(k+i)+\overline{B}(k+i)K\mathbf{x}(k-h_k+i)\right]\right]^T P\mathbf{x}(k)\\ &+\mathbf{x}^T(k)P\left[(\overline{A}(k)+\overline{B}(k)K-I)\mathbf{x}(k)\right] - \mathbf{x}^T(k)P\left[\overline{B}K\sum_{i=-h_k}^{-1}\left[(\overline{A}(k+i)-I)\mathbf{x}(k+i)+\overline{B}(k+i)K\mathbf{x}(k-h_k+i)\right]\right]\\ &= \Delta X_1 + \mathbf{x}^T(k)\left[\overline{A}(k)+\overline{B}(k)K\right]^T P\mathbf{x}(k) + \mathbf{x}^T(k)P\left[\overline{A}(k)+\overline{B}(k)K\right]\mathbf{x}(k)\\ &-2\left[\overline{B}K\sum_{i=-h_k}^{-1}\left[(\overline{A}(k+i)-I)\mathbf{x}(k+i)+\overline{B}(k+i)K\mathbf{x}(k-h_k+i)\right]\right]^T P\mathbf{x}(k)\\ &\le \mathbf{x}^T(k)\Big\{(A+BK-I)^T P + P(A+BK-I) + P(r_1 L_A L_A{}^T + r_2 L_B L_B{}^T)P + r_1^{-1}E_A{}^T E_A + r_2^{-1}K^T E_B{}^T E_B K\\ &+HP\left[BQB^T + r_3 L_B L_B{}^T + BQE_B{}^T(r_3 I - E_B Q E_B{}^T)^{-1}EQB^T\right]P\Big\}\mathbf{x}(k) + \Delta X_1 + \Delta X_2 + \Delta X_3\ ,\end{aligned} \tag{5.18}$$

where $Q \ge KP^{-1}K^T$, $P^{-1} - P_1 - P_2 \ge 0$, and

$$\Delta X_2 = \sum_{i=-h_k}^{-1} \mathbf{x}^T(k+i)\left[(\overline{A}(k+i)-I)^T P_1^{-1}(\overline{A}(k+i)-I)\right]\mathbf{x}(k+i),$$
$$\Delta X_3 = \sum_{i=-h_k}^{-1} \mathbf{x}^T(k-h_k+i)\left[K^T\overline{B}^T(k+i)P_2^{-1}(k+i)\overline{B}(k+i)K\right]\mathbf{x}(k-h_k+i). \tag{5.19}$$

For ΔW, from (5.13) we obtain

$$\begin{aligned}\Delta W &= h_k\mathbf{x}^T(k)\left[(\overline{A}^T(k)-I)P_1^{-1}(\overline{A}(k)-I)\right]\mathbf{x}(k) + h_k\mathbf{x}^T(k)\left[K^T\overline{B}^T(k+h_k)P_2^{-1}\overline{B}(k+h_k)K\right]\mathbf{x}(k)\\ &- \sum_{i=-h_k}^{-1}\mathbf{x}^T(k+i)\left[(\overline{A}^T(k+i)-I)P_1^{-1}(\overline{A}(k+i)-I)\right]\mathbf{x}(k+i)\\ &- \sum_{i=-h_k}^{-1}\mathbf{x}^T(k-h_k+i)\left[K^T\overline{B}^T(k+i)P_2^{-1}(k+i)\overline{B}(k+i)K\right]\mathbf{x}(k-h_k+i)\ .\end{aligned} \tag{5.20}$$

Applying Lemma 5-1(2) and Assumption 5-2, we obtain

$$\begin{aligned}\Delta W \le \mathbf{x}^T(k)\Big\{&H\Big[(A-I)^T(P_1 - r_4L_AL_A^T)^{-1}(A-I) + r_4^{-1}E_A^TE_A\\ &+ K^TB^T(P_2 - r_5L_BL_B^T)^{-1}BK + r_5^{-1}K^TE_B^TE_BK\Big]\Big\}\mathbf{x}(k) - \Delta X_2 - \Delta X_3\ .\end{aligned} \tag{5.21}$$

For ΔS, from (5.14) we obtain

$$\begin{aligned}\Delta S &= S(\mathbf{x},0) + \sum_{i=-k-1}^{-1}\mathbf{x}^T(k+1+i)P\mathbf{x}(k+1+i) + \sum_{i=-k-1}^{-1}\left[\mathbf{x}(k+1+i)-\mathbf{x}(k+2+i)\right]^T P\left[\mathbf{x}(k+1+i)-\mathbf{x}(k+2+i)\right]\\ &-S(\mathbf{x},0) - \sum_{i=-k}^{-1}\mathbf{x}^T(k+i)P\mathbf{x}(k+i) - \sum_{i=-k}^{-1}\left[\mathbf{x}(k+i)-\mathbf{x}(k+1+i)\right]^T P\left[\mathbf{x}(k+i)-\mathbf{x}(k+1+i)\right]\\ &= \mathbf{x}^T(k)P\mathbf{x}(k) - \left[\mathbf{x}(k)-\mathbf{x}(k+1)\right]^T P\left[\mathbf{x}(k)-\mathbf{x}(k+1)\right] \le \mathbf{x}^T(k)(P+P)\mathbf{x}(k) - \Delta X_1\end{aligned} \tag{5.22}$$

Substituting (5.18–5.22) into (5.16), we obtain

$$\Delta V(\mathbf{x},k) = \Delta X + \Delta W + \Delta S \le \mathbf{x}^T(k) R(P,P_1,P_2,P_3,Q,K,H,r_i)\mathbf{x}(k). \tag{5.23}$$

Thus from (5.7–5.9) and (5.23), we have

$$\Delta V(\mathbf{x},k) \le \mathbf{x}^T(k) R(P,P_1,P_2,P_3,Q,K,H,r_i)\mathbf{x}(k) < 0\,. \tag{5.24}$$

The above sufficient condition for the existence of guaranteed controllers is equivalent to the solvability of a system with the following LMIs.

Theorem 5-2: Given an $H > 0$, the system (5.1–5.3) is robustly closed-loop stable through a control input of (5.6) for any time delay length parameter h_k satisfying $h_k \le H$ if there exist real symmetric positive-definite matrices Y, P_1, P_2, and Q, a matrix Z, and scalars $r_i > 0$, $i = 1, \ldots, 5$, such that the following LMIs hold:

$$\begin{bmatrix} M_{11}(Y,Z) & M_{12}(Y,Z) & M_{13} & M_{14}(Y,Z,H) \\ M_{12}{}^T(Y,Z) & M_{22} & 0 & 0 \\ M_{13}{}^T & 0 & M_{33} & 0 \\ M_{14}{}^T(Y,Z,H) & 0 & 0 & M_{44} \end{bmatrix} < 0 \tag{5.25}$$

$$\begin{bmatrix} Q & Z \\ Z^T & Y \end{bmatrix} \ge 0 \tag{5.26}$$

$$Y - P_1 - P_2 \ge 0\,, \tag{5.27}$$

where

$$M_{11}(Y,Z) = YA^T + AY + Z^T B^T + BZ + HBQB^T + r_1 L_A L_A{}^T + r_2 L_B L_B{}^T + H r_3 L_B L_B{}^T \tag{5.28}$$

$$M_{12}(P,K) = \begin{bmatrix} YE_A^T & Z^T E_B^T \end{bmatrix} \tag{5.29}$$

$$M_{13} = HBQE_B^T \tag{5.30}$$

$$M_{14}(Y,Z,H) = H\begin{bmatrix} Y(A^T - I) & YE_A^T & Z^T B^T & Z^T E_B^T \end{bmatrix} \tag{5.31}$$

$$M_{22} = -\mathrm{diag}\{r_1 I, r_2 I\} \tag{5.32}$$

$$M_{33} = -H(r_3 I - E_B Q E_B^T) \tag{5.33}$$

$$M_{44} = -H\,\mathrm{diag}\{P_1 - r_4 L_A L_A^T, r_4 I, P_2 - r_5 L_B L_B^T, r_5 I\}, \tag{5.34}$$

and a stabilizing controller is given by $\mathbf{u}(k) = ZY^{-1}\mathbf{x}(k)$.

Proof: Define the new variables Y and Z in (5.25–5.27) as

$$Y = P^{-1}, Z = KY. \tag{5.35}$$

Multiplying the both sides of the inequality (5.25) by Y, and then by Schur complements, we obtain that the conditions in (5.7–5.9) are equivalent to the LMIs (5.25–5.27).

5.2.2 Robust H_∞ Control

Assuming $\mathbf{w}(k) \neq 0$, we also present a sufficient condition as the following theorem for memoryless state-feedback controllers for the uncertain system (5.1–5.3) to be robustly stable with a prescribed level of disturbance attenuation.

Theorem 5-3: Given an $H > 0$ and $\gamma > 0$, the system (5.1–5.3) is robustly closed-loop stable with a disturbance attenuation γ with a control input of (5.6) for any time delay length parameter h_k,

satisfying $h_k \le H$ if there exist real symmetric positive-definite matrices Y, P_1, P_2, P_3, and Q, a matrix Z, and scalars $r_i > 0$, $i = 1, \ldots, 5$, such that the following LMIs hold:

$$\begin{bmatrix} M_{11}(Y,Z) & M_{12}(Y,Z) & M_{13} & M_{14}(Y,Z,H) & M_{15}(Y) \\ M_{12}{}^{T}(Y,Z) & M_{22} & 0 & 0 & 0 \\ M_{13}{}^{T} & 0 & M_{33} & 0 & 0 \\ M_{14}{}^{T}(Y,Z,H) & 0 & 0 & M_{44} & 0 \\ M_{15}{}^{T}(Y) & 0 & 0 & 0 & M_{55} \end{bmatrix} < 0 \tag{5.36}$$

$$\begin{bmatrix} Q & Z \\ Z^T & Y \end{bmatrix} \ge 0 \tag{5.37}$$

$$Y - P_1 - P_2 - P_3 \ge 0\,, \tag{5.38}$$

where M_{11}, M_{12}, M_{13}, M_{14}, M_{22}, M_{33}, and M_{44} are as defined in (5.28–5.34), and

$$M_{15} = \begin{bmatrix} YE^T & B_w & 0 \end{bmatrix} \tag{5.39}$$

$$M_{55} = -\begin{bmatrix} I & 0 & 0 \\ 0 & \gamma^2 I & HB_w{}^{T} \\ 0 & HB_w & HP_3 \end{bmatrix}, \tag{5.40}$$

and a guaranteed controller is given by $\mathbf{u}(k) = ZY^{-1}\mathbf{x}(k)$.

Proof: Apply the control input (5.6) to the system (5.1–5.3), the closed-loop transfer function T_{zw} from the disturbance input $\mathbf{w}(k)$ to the controlled output $\mathbf{z}(k)$ is given by

$$T_{zw} = E(zI - \overline{A} - \overline{B}Kz^{-h_k})^{-1} B_w\,, \tag{5.41}$$

From (5.5) and (5.41), and by the result of Theorem 5-1, Lemma 5-1, and Schur complements, we obtain Theorem 5-3.

5.2.3 Robust Parameter Optimization

The problem of control-parameter optimization, like finding the largest H for a given γ or the smallest γ for a given H, can be easily solved using standard LMI approaches by Theorem 5-3. For instance, the largest H obtainable from Theorem 5-3 which ensures that the system (5.1–5.3) is robustly stabilizable with the disturbance attenuation γ can be determined by solving the following quasi-convex optimization problem:

LMIs: (5.36–5.38)

Objective: Maximize H subject to

$$H > 0, Y > 0, P_1 > 0, P_2 > 0, P_3 > 0, Q > 0, Z > 0, r_i > 0, i = 1, \ ..., \ 5.$$

On the other hand, the smallest γ obtainable from Theorem 5-3 which ensures that the system (5.1–5.3) with a given H is robustly stabilizable can be determined by solving the following quasi-convex optimization problem:

LMIs: (5.36–5.38)

Objective: Minimize γ^2 subject to

$$Y > 0, P_1 > 0, P_2 > 0, P_3 > 0, Q > 0, Z > 0, r_i > 0, i = 1, \ ..., \ 5 .$$

5.2.4 Dynamic Controller Design

In this approach, we design a dynamic controller to solve the robust stabilization problem. For the NCS shown in Fig. 5-1, since the controller should have network connection capability, it usually has some memory to store the previous control data. Here we give a sufficient condition for the robust stability of NCSs.

Assuming $w(k) = 0$, the discrete-time analogue of Artstein transform [31] for the system (5.1–5.3) is given by

$$\tilde{\mathbf{x}}(k) = \mathbf{x}(k) + \sum_{i=-h_k}^{-1} \overline{A}(k)^{-(h_k+i+1)} \overline{B}(k)\mathbf{u}(k+i)\,. \tag{5.42}$$

The derivation of this discrete-time Artstein transform is given in the Appendix A.

Lemma 5-2: Let ($\mathbf{x}(k)$, $\mathbf{u}(k)$) be a solution (admissible pair) for (5.1–5.3), defined by initial condition ($\mathbf{x}(0)$, $\mathbf{u}_0(.)$). Then $(\tilde{\mathbf{x}}(k), \mathbf{u}(k))$ with $\tilde{\mathbf{x}}(k)$ defined by (5.42) is a solution (admissible pair) for the system

$$\tilde{\mathbf{x}}(k+1) = \overline{A}(k)\tilde{\mathbf{x}}(k) + \overline{A}(k)^{-h_k} \overline{B}(k)\mathbf{u}(k) \tag{5.43}$$

with the initial condition $(\tilde{\mathbf{x}}(0), \mathbf{u}_0(.))$. Conversely, let $(\tilde{\mathbf{x}}(k), \mathbf{u}(k))$ be a solution of (5.43) defined by some initial condition $(\tilde{\mathbf{x}}(0), \mathbf{u}_0(.))$. Then given some $u_0(.)$ defined on $\left[-h_0 \quad 0\right)$ and taking

$$\mathbf{x}(0) = \tilde{\mathbf{x}}(0) - \sum_{i=-h_0}^{-1} \overline{A}(0)^{-(h_0+i+1)} \overline{B}\mathbf{u}(i)\,, \tag{5.44}$$

the solution for (5.1) by the initial condition (5.44) is given by

$$\mathbf{x}(k) = \tilde{\mathbf{x}}(k) - \sum_{i=-h_k}^{-1} \overline{A}(k)^{-(h_k+i+1)} \overline{B}(k)\mathbf{u}(k+i)\,. \tag{5.45}$$

The proof of this result is straightforward.

Lemma 5-3: Let $\mathbf{u}(k) = F\tilde{\mathbf{x}}(k)$ be a feedback controller stabilizing the system (5.43). Then the controller

$$\mathbf{u}(k) = F\left[\mathbf{x}(k) + \sum_{i=-h_k}^{-1} \overline{A}(k)^{-(h_k+i+1)} \overline{B}(k)\mathbf{u}(k+i)\right] \tag{5.46}$$

is stabilizing the system (5.1–5.3).

The proof is straightforward and relies entirely on Lemma 5-2. Since the admissible uncertainties are unknown, the dynamic controller is given as

$$\hat{\mathbf{u}}(k) = F\left[\mathbf{x}(k) + \sum_{i=-h_k}^{-1} A^{-(h_k+i+1)} B\hat{\mathbf{u}}(k+i)\right]. \tag{5.47}$$

This compensator can be constructed by steps.

The corresponding nominal state equations of the transformed system is

$$\begin{aligned} \hat{\mathbf{x}}(k+1) &= A\hat{\mathbf{x}}(k) + A^{-h_k} B\hat{\mathbf{u}}(k) \\ \hat{\mathbf{u}}(k) &= F\hat{\mathbf{x}}(k). \end{aligned} \tag{5.48}$$

Thus the augmented system equation is

$$\begin{bmatrix} \widetilde{\mathbf{x}}(k+1) \\ \hat{\mathbf{x}}(k+1) \end{bmatrix} = \begin{bmatrix} \overline{A}(k) & \overline{A}(k)^{-h_k} \overline{B}F \\ 0 & A + A^{-h_k} BF \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}(k) \\ \hat{\mathbf{x}}(k) \end{bmatrix}. \tag{5.49}$$

Theorem 5-4: Given an $H > 0$, the system (5.1–5.3) is robustly closed-loop stable through a dynamic control input of form (5.47) for any time delay length parameter h_k satisfying $h_k \le H$ if the largest singular value of $\begin{bmatrix} \overline{A}(k) & \overline{A}(k)^{-H} \overline{B}F \\ 0 & A + A^{-H} BF \end{bmatrix}$ is less than 1.

The proof is straightforward.

5.2.5 Simulation and Experimental Verification

To verify the feasibility of the theorems developed in this section, we use the NCS test bed constructed in Chapter II again. The discrete-time model of the ball maglev system is

$$\begin{aligned}\mathbf{x}(k+1) &= \begin{bmatrix} 1 & 0 \\ 0.006 & 1 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} 0.003 \\ 0.000009 \end{bmatrix}\mathbf{u}(k) \\ \mathbf{z}(k) &= \begin{bmatrix} 0 & -1.6233 \end{bmatrix}\mathbf{x}(k)\ ,\end{aligned} \tag{5.50}$$

where (5.50) is the discrete-time state-space model of (2.2) with the sampling period h as 3 ms. Assume there is no disturbance input and the model uncertainties are set to be

$$L_A = \begin{bmatrix} 0.01 & 0 \\ 0.001 & 0.01 \end{bmatrix}, L_B = \begin{bmatrix} 0.0001 \\ 0 \end{bmatrix}, E_A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, E_B = 1\,. \tag{5.51}$$

First, based on the model (5.50–5.51), the upper bound of time delay that the maglev system can accommodate need to be found. Applying the robust control result of Theorem 5-2 to the above system (5.50–5.51) and solving LMIs (5.25–5.27), the upper bound of H was found to be 1.62. In other words, for any time-delay τ_k satisfying $\tau_k \le 1.62 \times 3 = 4.86$ ms, the system (5.50) is robustly closed-loop stabilizable. A simulation of this ball maglev system using Matlab Simulink is implemented to verify this upper bound of time delay. The block diagram of this simulation is presented in Appendix B.1. Fig. 5-2 shows the simulation result of system 1-mm step responses (starting from 0.25 s) with different time-delay upper bounds. In Fig. 5-2 (a), no time delay was introduced, in Fig. 5-2 (b), uniform time delays with upper bound of 4.86 ms were introduced, and in Fig. 5-3(c), time delays as long as 6.5 ms were introduced.

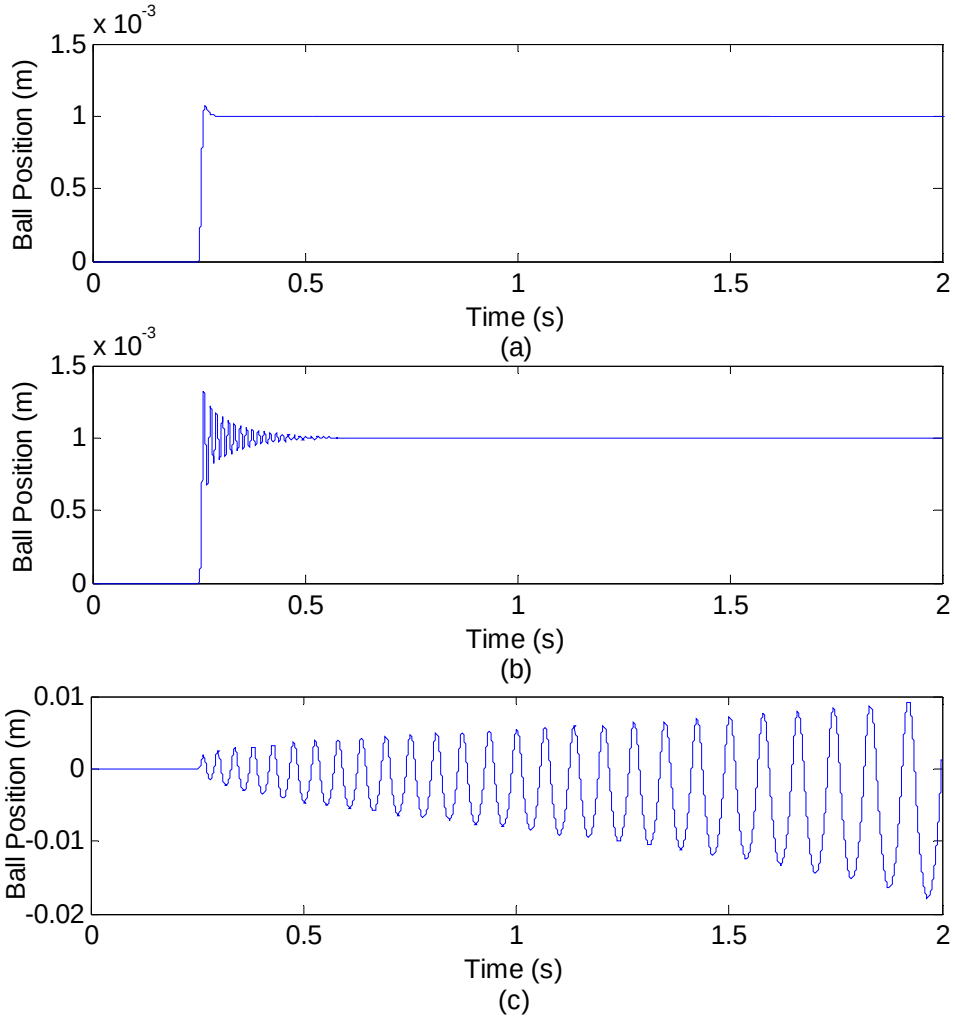


Fig. 5-2. Simulation result of system step responses with (a) no time delay, (b) uniform time delays with upper bound of 4.86 ms, and (c) time delays as long as 6.5 ms

As evidenced from Fig. 5-2 (b), the ball maglev system went to stable after several oscillations, thus it can accommodate time delays less than 4.86 ms. It worths mention that the system could be stable when time delay is longer than 4.86 ms, however, the oscillations in Fig. 5-2 (b) indicates that the system performance degrades and the system would become unstable when time delays longer than 4.86 ms are introduced. The time delay kept increasing during the

simulation, when time delay longer than 6 ms occurred, the system could not maintain its stability as shown in Fig. 5-2(c). Thus, the simulation results verify that the theorem 5-2 can be used to determine the upper bound of the time delay even though it might be a little conservative.

Then we measured the real round-trip time delay induced by the Ethernet in our lab. Refer to Fig. 4-1, the average round-trip time delay is about 230 μs, its standard deviation is about 200 μs, and the maximum time delay is about 3.4 ms that is less than 4.86 ms. Thus the maglev system is supposed to be stabilizable with a control loop closed over the Ethernet. The response of the ball position with control loop closed over the Ethernet LAN in our lab is shown in Fig. 5-3. The ball maintained its equilibrium position and did not fall down. The NCS test bed is stabilized even with the present of the network-induced time delays shown in Fig. 4-1. Refer to Figs. 2-11 and 2-12 for the other system performances with control loop closed over the Ethernet LAN.

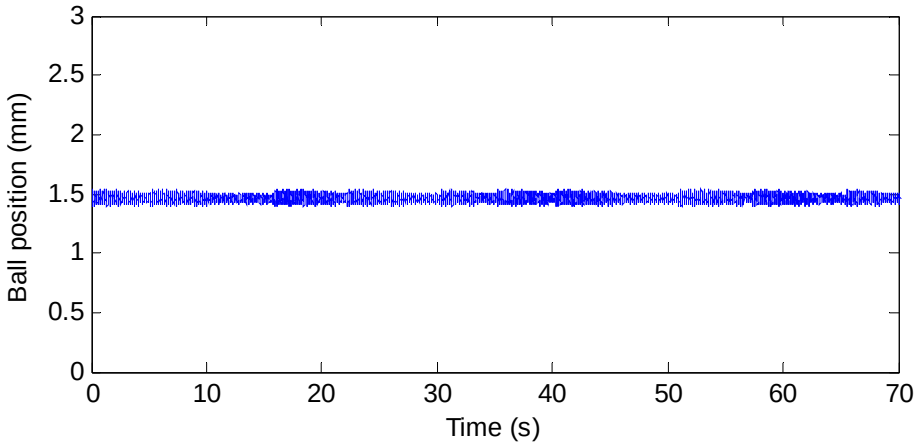


Fig. 5-3. System response with control loop closed over the Ethernet

To verify the upper bound of time delay, more experiments were conducted. As proposed in Chapter IV, the time-delay problem and packet-loss problem encountered in real-time

controlling the ball maglev system can be dealt with uniformly. In Chapter II, it was mentioned that the actuator needs control data within 1.42 ms after the sensor sampling. To introduce longer time delays, we introduced packet losses. One sensor data packet loss was introduced once every 1.8 s at the controller node, which is equivalent to introducing time delays as long as 4.42 ms. The previous control data was still used when packet loss occurred. Since the upper bound of the time delay that the ball maglev system can accommodate is found to be 4.86 ms. The system is expected to be stable. The system response of ball position is shown in Fig. 5-4. The NCS test setup maintained its stability successfully with periodic 0.4 mm peak-to-peak spikes in the ball position when packet losses occurred.

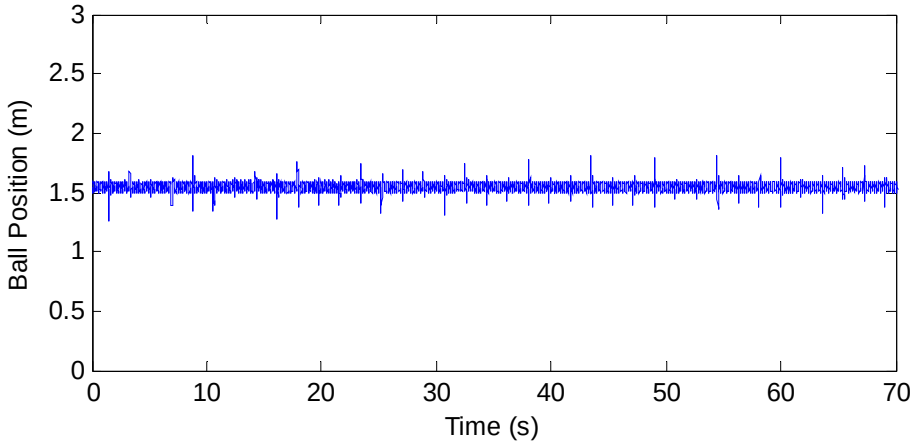


Fig. 5-4. System response with one packet loss (4.42 ms-long time delay) occurring every 1.8 s

To introduce time delay longer than 6 ms, we introduced 2 consecutive packet losses after 9 s, which is equivalent to introducing time delays as long as 7.42 ms. The system response is shown in Fig. 5-5. In this figure, the zero value of the vertical axis denotes that the system lost its stability and that the ball could not maintain its equilibrium position and fell down.

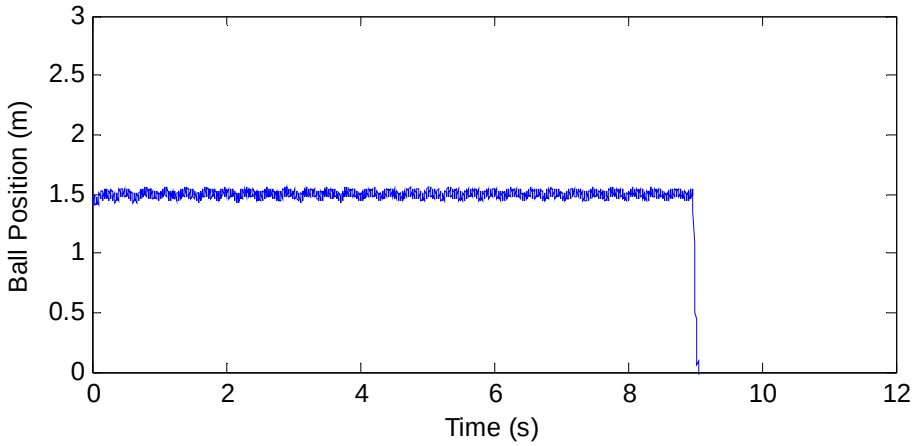


Fig. 5-5. System response with 2 successive packet losses (7.42 ms-long time delay) occurring after 9 s

Thus from Figs. 5-4 and 5-5, it can be concluded that the upper bound of time delay this ball maglev system can accommodate is about 4.42 ms , which is very close to the theoretic result 4.86 ms calculated from Theorem 5-2.

Another numerical example is presented to demonstrate and verify the robust control methodology proposed in this section. Consider NCS shown in Fig. 5-1 with parameter uncertainties and time delay described by the following discrete-time state equations

$$\begin{aligned} \mathbf{x}(k+1) &= (A + A_u)\mathbf{x}(k) + B_w\mathbf{w}(k) + (B + B_u)\mathbf{u}(k - h_k) \\ \mathbf{z}(k) &= E\mathbf{x}(k) \ , \end{aligned} \tag{5.52}$$

where sampling period h is 0.01 s and

$$A = \begin{bmatrix} 1 & 0.05 \\ 0.01 & 0 \end{bmatrix}, B = \begin{bmatrix} 0.002 \\ 0.01 \end{bmatrix}, B_w = \begin{bmatrix} 0.001 \\ 0 \end{bmatrix}, E = \begin{bmatrix} 1 & 0 \end{bmatrix},$$
$$L_A = \begin{bmatrix} 0 & 0 \\ 0.001 & 0.001 \end{bmatrix}, L_B = \begin{bmatrix} 0.001 \\ 0 \end{bmatrix}, E_A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, E_B = 1.$$

The robust control designs for (5.52) can be performed as following.

1. For the robust stabilization problem, applying the robust control result of Theorem 5-2 to the above system (5.52) and solving LMIs (5.25–5.27), it was found that the upper bound of H is 8.46. In the other words, for any time-delay τ_k satisfying $\tau_k \le 8.46 \times 0.01 = 0.0846$ s, the system (5.52) is robustly closed-loop stabilizable.
2. For the robust H_∞ control problem, given the time-delay parameter h_k satisfying $h_k \le 8.46$ and disturbance attenuation $\gamma = 3$. Applying Theorem 5-3 to the above system (5.52) and solving LMIs (5.36–5.38), it was found that the memoryless state-feedback control gain is $K = \begin{bmatrix} -10.1 & -10.4 \end{bmatrix}$.
3. For the problem of robust-parameter optimization, there are two optimization problems:

(1) Finding the maximum H: Given $\gamma = 3$, applying the H_∞ control result of Theorem 5-3 to the system (5.52) and solving LMIs (5.36–5.38), it was found that the maximum H is 8.4. If given $\gamma = 0.5$, then the maximum allowed time-delay upper bound parameter H decreases to 2.6.

(2) Finding the minimum γ: Given $H = 1$, applying the H_∞ control result of Theorem 5-3 to the system (5.52) and solving LMIs (5.36–5.38), it was found that for time-delay parameter h_k satisfying $h_k \le 1$, the smallest value of γ is 0.2. If given $H = 8$, the smallest value of γ increases to 1.7.

The relation between the maximum H and the minimum γ is shown in Fig. 5-6 with a series of marked simulation results. As expected, the maximum H monotonically increases as the disturbance attenuation γ gets greater. However, the disturbance attenuation performance beyond a certain threshold ($\gamma \approx 2.3$) makes little difference, and H approaches 8.46, which is the maximum H without disturbance in the robust stabilization problem. The block diagram of this simulation is presented in Appendix B.3.

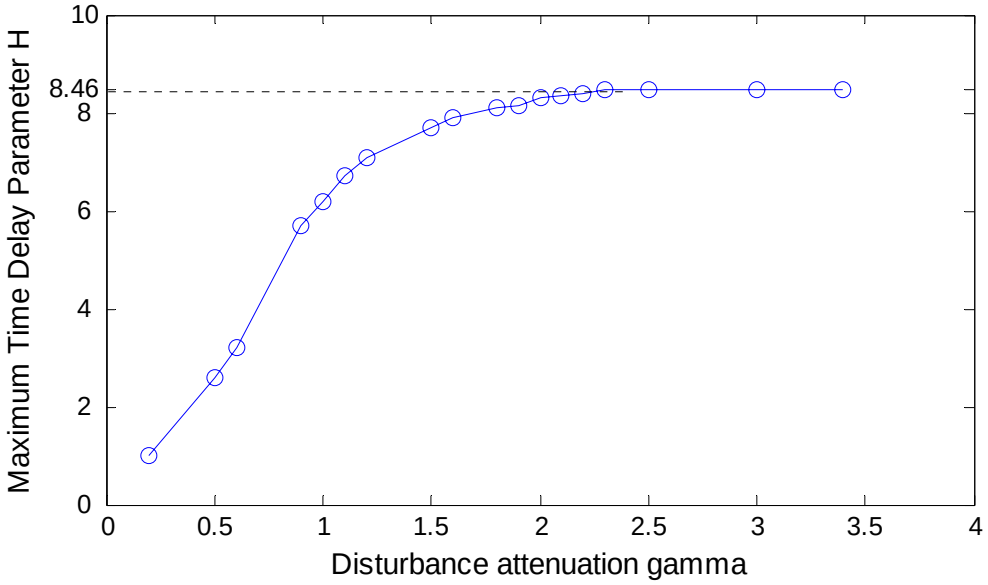


Fig. 5-6. Relation between the maximum H and the minimum γ

5.3 Optimal Co-Design Control of NCSs

In this section, an optimal co-design problem for NCSs is addressed and formulated. An important point to consider in the design of an NCS as shown in Fig. 3-1 is that the dynamic behavior of the distributed architecture largely depends on the characteristics and performance parameters of the underlying network, such as medium-access protocol, data-transmission rate, and available communication bandwidth. It is impossible to design an NCS without considering both views because the control performance of feedback control loops also depends on a new constraint—the limited bandwidth of the communication network. For example, the reduction of the sampling interval improves the control loop's performance [78]. However, a shorter sampling interval requires more network bandwidth to transmit more sensor data or control data, which increases the network traffic load. Thus, NCS design is a multidisciplinary field that includes the

optimal co-design of control systems and network communication systems. The network QoS should be analyzed together with specifying the control QoP before implementing the real-time control over networks.

5.3.1 Network Quality of Service

As discussed in Section 1.4, the main evaluation measures of the network QoS are time delay and packet loss statistics, network efficiency, and network utilization. These measures can be used to determine the capability of the network medium and to provide information to specify control parameters such as the sampling frequency.

The network efficiency is the ratio of the total transmitting time to the time used to send messages, including queuing time, block time, and so on [57]. If network efficiency approaches one, it means that all time delay is due to the transmission delay. If network efficiency approaches zero, it means that the most of the time delay is due to message contention or collision.

The network utilization is defined as the ratio of the total time used to transmit data and the total running time, which is the sum of the ration of message transmission times and message periods of all devices [57]. If the network utilization approaches zero, there is network bandwidth available for other functionalities or control purposes. If the network utilization approaches one, the network becomes saturated, and it is difficult to increase the sampling rates of control devices or add more devices. Then either network bandwidth reallocation to reassign the traffic load or network redesign is needed. Thus when designing a real-time control over networks, the network QoS based on the above-mentioned measures needs to be evaluated. Refer to Section 1.4 and [57] for the detailed discussion and case studies of different network protocols. Refer to Chapter VI for a novel optimal network-bandwidth-allocation algorithm to dynamically

reallocate the bandwidth to optimize the QoC of the NCS.

5.3.2 Control Quality of Performance

There are several control specifications can be used to evaluate the control QoP such as phase margin, rising time, steady-state error, integral of the absolute value of the error (IAE), integral of the time multiplied by the absolute value of the error (ITAE), and so on.

The phase margin is the amount by which the phase of a loop transfer function exceeds -180^{o} when its magnitude equals one in the system Bode plot. From digital control theory, the phase lags due to discretization, $\Delta\Phi_s$, and time delay, $\Delta\Phi_d$ are presented as [78] [38]

$$\begin{aligned} \Delta\Phi_s &= \frac{\omega h}{2} \\ \Delta\Phi_d &= \omega\tau, \end{aligned} \tag{5.53}$$

where h is the sampling period and τ is the time delay.

To guarantee acceptable control performance such as response rapidness and smoothness, the "rule of thumb" for selecting the sampling frequency in digital control is that it should be 20~40 times as high as the closed-loop bandwidth ω_{bw} [78] [38], that is

$$20 \le \omega_s / \omega_{bw} \le 40, \tag{5.54}$$

where $\omega_s = 2\pi / h$ is the sampling frequency, and ω_{bw} is the control-system closed-loop bandwidth defined as the frequency of the input at which the output is attenuated by 3 dB compare with the DC output [78]. Alternatively, the sampling periods are 4~10 per rise time t_r

[78], that is

$$4 \le t_r / h \le 10\,, \tag{5.55}$$

IAE and ITAE are two criteria generally used to evaluate control system performance. They are formulated in continuous-time as [78] and [38]

$$\begin{aligned} IAE &= \int_{t_0}^{t_f} |e| dt \quad \text{and} \\ ITAE &= \int_{t_0}^{t_f} t|e| dt. \end{aligned} \tag{5.56}$$

In discrete-time as

$$\begin{aligned} IAE &= \sum_{k=k_0}^{k_f} |e_k| \quad \text{and} \\ ITAE &= \sum_{k=k_0}^{k_f} k|e_k|\,, \end{aligned} \tag{5.57}$$

where t_0 (or k_0) and t_f (or k_f) are the initial and final times of the evaluation period and e is the system error defined as the error between the actual and reference trajectories.

5.3.3 NCS Design Chart

As proposed in [38], the system performance of an NCS is affected by network characteristics, such as data-transmission rate, time delays, data-packet losses, network

efficiency, and network utilization. Thus in an NCS design, there exists a system performance chart as shown in Fig. 5-7, which is a modification of Fig. 1 in [38]. As the sampling frequency gets higher, the traffic load becomes heavier in a bandwidth-limited network. Then the possibility of longer time delay or more data-packet loss increases. Thus there exist the optimal performance point γ and the minimum acceptable performance points α and β as denoted in Fig. 5-7.

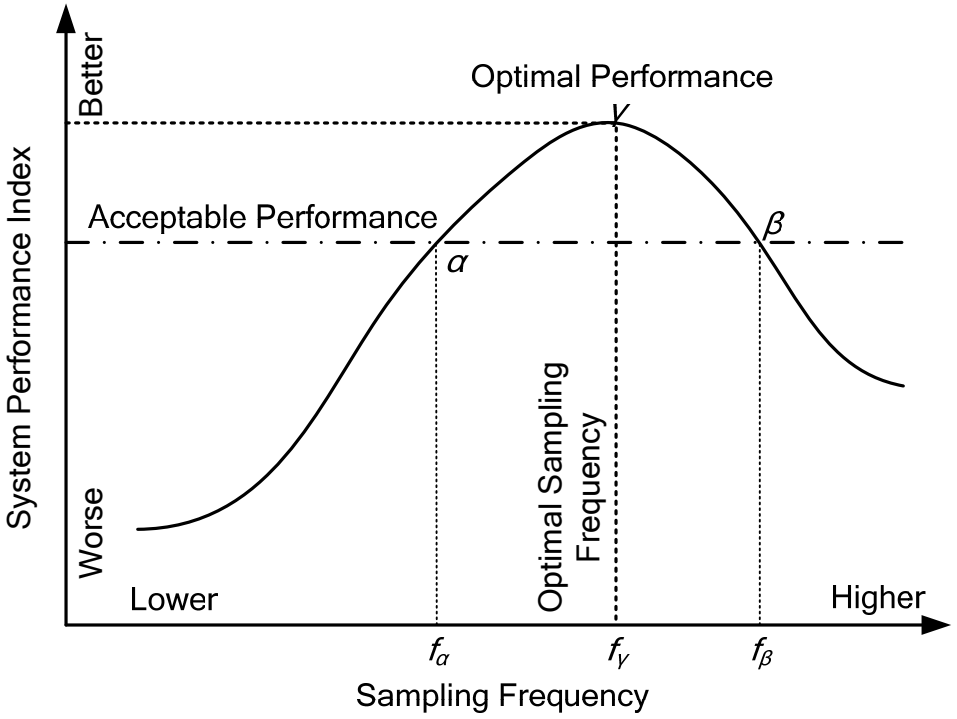


Fig. 5-7. System performance vs. sampling frequency. Modified after [38]

As proposed in [38], the worst, acceptable, and best regions of performance can be defined based on control-system specifications such as robust stability, system overshoot, steady-state error, and phase margin. The performance axis in Fig. 5-7 could be chosen to reflect a

subset of these metrics. Thus the points α, β, and γ can be determined by further investigating the control-system specifications and the characteristics and statistics of network-induced time delays and device processing time. Based on the work in [38], we define the acceptable performance as stability, i.e. the interval (f_α, f_β) in Fig. 5-7 is the working range of the sampling frequency with which the system can be stabilized. In the following section, we propose estimation method to determine this working range of the sampling frequency as a guideline of an NCS design.

5.3.4 Sampling Frequency Selection

1. The lowest allowable sampling frequency f_α

By Theorem 3-1 proposed in Chapter III, the upper bound of the time delay τ that the system can accommodate can be determined by using the standard LMI solver. Furthermore, if we know the network characteristics and statistics, such as the lower bound of the data transmission rate and the device processing time, then the lowest allowable sampling frequency at Point α in Fig. 5-7 can be estimated by

$$f_\alpha \approx 1/(\tau - T_d), \tag{5.58}$$

where $T_d = T_t + T_p$ is the network-induced time delay that includes the data-transmission time T_t and the device processing time T_p. T_t can be estimated as $T_t = L/R$ where R is the data rate in bytes per second for the current control application, and L is the average data-packet size in bytes. The device processing time T_p includes the time delays at the source and destination nodes. The processing time at the source node includes the preparing time T_{pre} (such as the time for AD or DA conversion) and the waiting time T_{wait}, and the processing time at the destination node

(such as the time for AD or DA conversion) is the post-processing time T_{post}. Thus the total device processing time T_p is given by

$$T_p = T_{pre} + T_{wait} + T_{post}. \tag{5.59}$$

To verify the sufficient stability condition derived in Chapter III, we used a numerical example (3.35) in Section 3.3. Let us consider this example again, and suppose the Ethernet is used as the communication medium to close the control loop. With the upper bound of time delay τ = 0.964 s, assuming T_p of 10 ms, the minimum data packet size L of 76 bytes, and the lower bound of data rate R of 100 kbps, the lowest allowable sampling frequency is obtained by (5.58) as $f_\alpha \approx 1 / (\tau - L/R - T_p) = 1 / (0.964 - (76 * 8) / (100 * 1024) - 0.01) \approx 1.05$ Hz.

2. Network bandwidth and optimal sampling frequency f_γ

Since the network is shared as a communication medium in an NCS, the network bandwidth may be limited for the control application. Point γ in Fig. 5-7 is the turn-around point of system performance vs. sampling frequency at which the network traffic begins to be saturated, that is, the network utilization approaches one. Therefore, f_γ can be estimated by considering the device processing time and the total-transmission-time T_{ttt} of all cyclic messages in the network applications. A good estimation suggested by the sufficient schedulability condition in [82] can be used for the estimation of the optimal sampling frequency at Point γ [38].

$$f_\gamma \approx 0.69 / (T_{ttt} + T_p), \tag{5.60}$$

where 0.69 is the maximum ratio of utilization to meet the sufficient schedulability condition for infinite messages, and

$$T_{ttt} = \sum_{i=1}^{n} T_{tran}^{i} = \sum_{i=1}^{n} l^{i} / R, \tag{5.61}$$

where n is the number of devices, T_{tran}^{i} is the transmission time of each data packet, and l^{i} is the data bit length of each packet.

Remark 5-1: In [38], this estimation was used to determine the maximum sampling frequency, since at Point γ, the network already begins to be saturated, it can be considered as the optimal sampling frequency point.

3. The highest allowable sampling frequency f_{β}

When the sampling frequency gets higher with a limited network bandwidth for the control application, the network traffic will be fully saturated and data-packet losses and longer time delays will occur more frequently. If the longest time delay is longer than τ, the system stability may be lost, i.e. $(f_{\beta}\tau L)/R = \tau - T_p$. Thus the highest allowable sampling frequency at Point β can be estimated by

$$f_{\beta} \approx (\tau - T_p)R/(\tau L). \tag{5.62}$$

For instance, let us consider the numerical example (3.35) again with the same parameters. Assuming T_p of 10 ms, the average data packet size L of 76 bytes, and the lower bound of the data rate for the current control application of 100 kbps, the highest allowable sampling frequency is obtained as $f_{\beta} \approx 100 * 1024 (0.964 – 0.01) / (0.964 * 76 * 8) \approx 166.67$ Hz

by (5.62). Thus considering the stability, the working range of the sampling frequency of the system (3.35) with the control loop closed over the network is (1.05 Hz, 166.67 Hz).

5.3.5 Optimal Controller Design

NCS performance improvement can be achieved in two ways. One is the minimization of device processing times and the improvement of network protocols to further guarantee the determinism of the data-transmission time as well as to reduce the end-to-end delays [38]. Refer to Chapters I and II for control network protocols selection and real-time operation system design for an NCS design. The designer should then pick a sampling frequency between points α and β (best at γ) in Fig. 5-7 with the performance criteria based on given design constraints.

The other way for performance improvement is through advanced optimal controller design that can overcome the uncertainty in an NCS and achieve the best QoC. Based on the discrete-time model (3.4) proposed in Chapter III, an optimal controller design with the consideration of both control and network parameters is formulated to improve the overall NCS performance with the following two cases.

1. Full-state-feedback case

Denote h^* as the optimal sampling period found in Section 5.3.4. We introduce a new augmented state variable $\mathbf{z}(i_k) = \left[\mathbf{x}^T(i_k) \quad \mathbf{u}^T(i_{k-1}) \quad \ldots \quad \mathbf{u}^T(i_{k-p})\right]^T \in R^{n+pm}$, then the augmented system can be expressed as follows

$$\begin{aligned} \mathbf{z}(i_{k+1}) &= \Phi(i_k)\mathbf{z}(i_k) + \Gamma(i_k)\mathbf{u}(i_k) + \Sigma\mathbf{v}(i_k), \\ \mathbf{y}(i_k) &= C_0\mathbf{x}(i_k) + \mathbf{w}(i_k), \end{aligned} \tag{5.63}$$

where

$$\Phi(i_k)=\begin{bmatrix} \Phi & B_{i_k}^1 & B_{i_k}^2 & \cdots & B_{i_k}^{p-1} & B_{i_k}^p \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & I_m & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & I_m & 0 \end{bmatrix}, \Gamma(i_k)=\begin{bmatrix} B_{i_{ki}}^0 \\ I_m \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \Sigma=\begin{bmatrix} I_m \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$C_0=\begin{bmatrix} C & 0 & \cdots & 0 \end{bmatrix},$$

$$\Phi=\Phi(h^*), \tag{5.64}$$

$$B_{i_k}^j=B_{i_k}^j(h^*),$$

I_m is the $m\times m$ identity matrix.

In this section, we consider the problem of minimizing the following cost function.

$$J(i_N)=E\left\{\mathbf{z}^T(i_N)Q_N\mathbf{z}(i_N)+\sum_{k=0}^{N-1}\left[\mathbf{z}^T(i_k)Q_0\mathbf{z}(i_k)+\mathbf{u}^T(i_k)R_0\mathbf{u}(i_k)\right]\right\} \tag{5.65}$$

where Q_N and Q_0 are symmetric positive-semi-definite, and R_0 is symmetric positive-definite.

Lemma 5-4: Let $E\{\cdot|y\}$ denote the conditional expectation given y. Assume that the function $f(x,y,u)=E\{l(x,y,u)|y\}$ has a unique minimum with respect to $u\in U$ for all $x\in X$ and $y\in Y$. Let $u^0(x,y)$ denote the value of u for which the minimum is achieved. Then we have

$$\min_{u(x,y)}E\{l(x,y,u)\}=E\{l(x,y,u^0(x,y))\}=E\left\{\min_{u(x,y)}E\{l(x,y,u)|y\}\right\}. \tag{5.66}$$

Proof: See [83].

Theorem 5-5: When the system has full state information, the control law that minimizes cost

function (5.65) is given by

$$\mathbf{u}(i_k) = -L(i_k)\left[\mathbf{x}^T(i_k) \quad \mathbf{u}^T(i_{k-1}) \quad \cdots \quad \mathbf{u}^T(i_{k-p})\right]^T, \tag{5.67}$$

where

$$\begin{aligned} L(i_k) &= \left[R_0 + E\left\{\Gamma^T(i_k)S(i_{k+1})\Gamma(i_k)\right\}\right]^{-1} E\left\{\Gamma^T(i_k)S(i_{k+1})\Phi(i_k)\right\}, \\ S(i_k) &= E\left\{\left[\Phi(i_k) - \Gamma(i_k)L(i_k)\right]^T S(i_{k+1})\left[\Phi(i_k) - \Gamma(i_k)L(i_k)\right]\right\} + L^T(i_k)R_0L(i_k) + Q_0, \\ S(i_N) &= Q_N. \end{aligned} \tag{5.68}$$

Proof: By repeatedly applying Lemma 5-4 to the cost function (5.65), we obtain the following Bellman equation:

$$J(i_k) = \min_u E\left\{\mathbf{z}^T(i_k)Q_0\mathbf{z}(i_k) + \mathbf{u}^T(i_k)R_0\mathbf{u}(i_k) + J(i_{k+1})\middle|\mathbf{z}(i_k)\right\}. \tag{5.69}$$

We will prove that the solution to (5.69) is in the form of

$$J(i) = E\left\{\mathbf{z}^T(i)S(i)\mathbf{z}(i)\right\} + \alpha(i), \tag{5.70}$$

where $S(i)$ is a symmetric matrix and denotes the cost to go at time i, and $\alpha(i)$ is the part of the cost function that is not affected by the control. The initial value at $i = i_N$ is

$$J(i_N) = \min_u E\left\{\mathbf{z}^T(i_N)Q_N\mathbf{z}(i_N)\middle|\mathbf{z}(i_N)\right\} = E\left\{\mathbf{z}^T(i_N)Q_N\mathbf{z}(i_N)\right\} \tag{5.71}$$

with $S(i_N) = Q_N$. Thus (5.69) is the solution when $i = i_N$. If the solution holds when $i = i_{k+1}$ and we can prove that it still holds when $i = i_k$, we can prove that (5.69) is the solution to (5.69) by

mathematical induction.

$$
\begin{aligned}
J(i_k) &= \min_{u(i_k)} E\left\{\mathbf{z}^T(i_k)Q_0\mathbf{z}(i_k) + \mathbf{u}^T(i_k)R_0 u(i_k) + J(i_{k+1}) \middle| \mathbf{z}(i_k)\right\} \\
&= \min_{u(i_k)} E\left\{\mathbf{z}^T(i_k)Q_0\mathbf{z}(i_k) + \mathbf{u}^T(i_k)R_0\mathbf{u}(i_k) + E\left\{\left[\Phi(i_k)\mathbf{z}(i_k) + \Gamma(i_k)\mathbf{u}(i_k)\right]^T S(i_{k+1})\left[\Phi(i_k)\mathbf{z}(i_k) + \Gamma(i_k)\mathbf{u}(i_k)\right]\right\} \middle| \mathbf{z}(i_k)\right\} \\
&\quad + tr\left[\Sigma^T S(i_{k+1)})\Sigma R_1\right] + \alpha(i_{k+1}).
\end{aligned}
\tag{5.72}
$$

Minimizing (5.72) with respect to $\mathbf{u}(i_k)$ gives the control law (5.67), and

$$
\begin{aligned}
J(i_k) &= E\left\{\mathbf{z}^T(i_k)\left\{Q_0 + L(i_k)^T R_0 L(i_k) + E\left\{\left[\Phi(i_k) - \Gamma(i_k)L(i_k)\right]^T S(i_{k+1})\left[\Phi(i_k) - \Gamma(i_k)L(i_k)\right]\right\}\right\}\mathbf{z}(i_k)\right\} \\
&\quad + \mathrm{tr}(\Sigma^T S(i_{k+1)})\Sigma R_1) + \alpha(i_{k+1}) \\
&= E\left\{\mathbf{z}^T(i_k)S(i_k)\mathbf{z}(i_k)\right\} + \alpha(i_k),
\end{aligned}
\tag{5.73}
$$

where

$$
\begin{aligned}
S(i_k) &= E\left\{\left[\Phi(i_k) - \Gamma(i_k)L(i_k)\right]^T S(i_{k+1})\left[\Phi(i_k) - \Gamma(i_k)L(i_k)\right]\right\} + L^T(i_k)R_0 L(i_k) + Q_0, \\
\alpha(i_k) &= \mathrm{tr}\left[\Sigma^T S(i_{k+1})\Sigma R_1\right] + \alpha(i_{k+1}), \\
L(i_k) &= \left[R_0 + E\left\{\Gamma^T(i_k)S(i_{k+1})\Gamma(i_k)\right\}\right]^{-1} E\left\{\Gamma^T(i_k)S(i_{k+1})\Phi(i_k)\right\}.
\end{aligned}
\tag{5.74}
$$

Since Q_0 and R_0 are symmetric, $S(i_k)$ is symmetric, thus the solution to the Bellman equation (5.69) is (5.70), which completes the proof.

2. Estimated-state-feedback control

When full state information is unavailable for the controller, a conventional way is to estimate unknown states with Kalman filters. Then the method developed above can also be

applied in this case. A key issue with respect to this method is to verify whether the separation principle holds. Refer to [84] for a design idea of estimated-state feedback and output feedback.

5.3.6 Case Study

In this section, we provide a case study of the ball maglev NCS test bed and illustrate the co-design of network and control system parameters. The system model of ball maglev system is presented as transfer function in (2.2). The state-space model is given as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \mathbf{u}(t), \\ \mathbf{y}(t) &= \begin{bmatrix} 0 & -1.6233 \end{bmatrix} \mathbf{x}(t). \end{aligned} \tag{5.75}$$

First we collected the network traffic information of the Ethernet LAN and the device processing time for data processing. A 76-byte-long data packet was sent from the plant PC to the controller PC and then came back from the controller PC to the plant PC. The mean value of the round-trip time delay was 230 μs (refer to Fig. 4-1), and we estimated that the data-transmission rate available for the test bed was about 2 Mbps. Based on the real-time operating environment (Linux RTAI) implemented on both the controller PC and the plant PC, the device processing time for A/D and D/A conversions and control calculation could also be measured. For our NCS test bed, the device processing time T_p was estimated to be 800 μs based on the above measurement.

Applying Theorem 3-1 to the system model (5.75) and solving the LMI (3.8), we found that the upper bound of the time delay for (5.75) with a full-state feedback controller would be 0.051 s. By the discussion in Section 5.3.4, the working range of the sampling frequency can be

estimated. From (5.58), we can estimate the minimum sampling frequency at Point α for the ball maglev set up as $f_{\min} \approx 1/(\tau - T_t - T_p) = 1/(0.051 - 0.00023 - 0.0008) \approx 20$ Hz. Ethernet packets with the minimum size (76 bytes) were used to carry the sensor data and the control data. Then from (5.62), we can estimate the maximum sampling frequency at Point β as $f_{\max} \approx (\tau - T_p)R/(\tau L) = (0.051 - 0.0008) * 2 * 1048576 / (0.051 * 76 * 8) \approx 3.4$ kHz. Thus, based on the control and network co-design consideration, the allowable working range of the sampling frequency for this NCS test bed is from (20 Hz, 3.4 kHz).

Since the ball maglev system has only one position sensor, full-state feedback is not available. Several output feedback controllers were designed to obtain the best control QoP. However, in simulation, full-state feedback controller can be designed to control the ball maglev simulation model. Several simulations of the ball maglev system (5.75) with different sampling frequencies were performed using Matlab Simulink to verify the maximum constant time delay that the system can accommodate. The simulation block diagram is presented in Appendix B.2. A plot of the maximum allowable constant time delay with respect to various sampling frequencies is shown in Fig. 5-8. Fig. 5-8 shows a general trend that the maximum time delay that can be accommodated at a higher sampling frequency is less than that at a lower sampling frequency. As the sampling frequency decreases, more time is available in the real-time control routine without frame overruns. Thus, the maximum allowable time delay keeps increasing with lowering of the sampling frequency. However, at a very low sampling frequency again the maximum allowable time delay decreases. This trend can be accounted for the poorer stability of the system at a low sampling frequency. Thus, there is an optimal sampling frequency for which the system can accommodate the maximum allowable constant time delay without affecting the system stability. It is worthy mentioning that the Fig. 5-8 is a form of system design chart which indeed follows the shape of the plot given in Fig. 5-7.

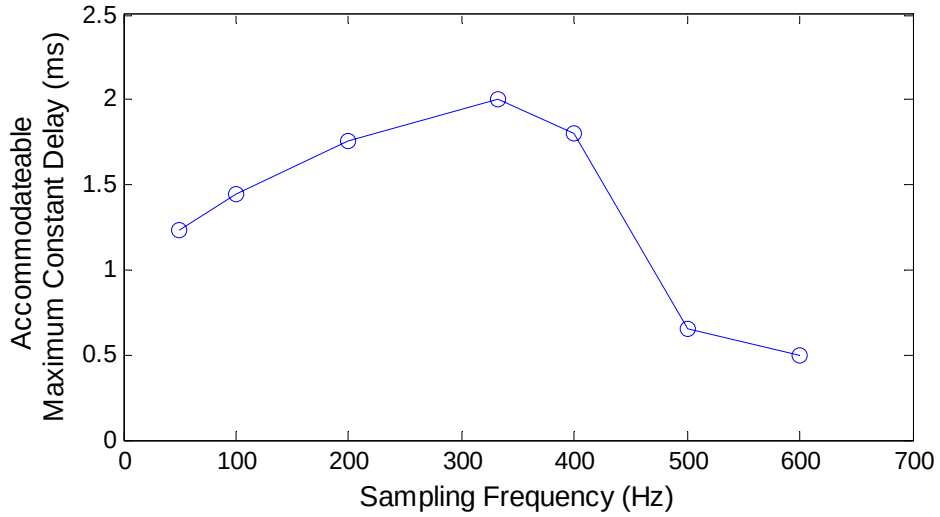


Fig. 5-8. Maximum allowable constant time delay vs. sampling frequencies

From the simulation results in Fig. 5-8, the best sampling frequency was found to be about 333.3 Hz. Fig. 5-8 verifies the choice of choosing 3 ms as sampling period in the first beginning. The discrete-time state space model of the ball maglev system with the 3-ms sampling period is presented in (5.50). A practical output feedback controller in discrete-time for the ball maglev system was designed as [22]:

$$u(i_k) = 0.78u(i_{k-1}) + 0.13u(i_{k-2}) - 18.92y(i_k) + 33.19y(i_{k-1}) - 15.06y(i_{k-2}) + 0.24. \quad (5.76)$$

With the ball maglev test bed and the 100-Mbps Ethernet LAN in our lab, a real-time NCS shown in Fig. 2-8 is successfully constructed in Chapter II.

5.4 Summary

Advanced NCS control design problems were investigated in this chapter. First we

formulated robust control problems for an NCS with norm-bounded parameter uncertainties. Robust stabilization, robust H_∞ control, and control-parameter optimization problems for NCSs were addressed. Delay-dependent methods of designing linear memoryless state-feedback controllers and dynamic state-feedback controllers to solve robust control problems were presented. Numerical examples were worked out to illustrate the presented robust control methodologies. Simulation and experiment results of the ball maglev test bed also verified the feasibility of these robust control methodologies. A similar LMI-based robust stabilization problem of the NCS with time-delay uncertainties only was proposed in [85] and the network induced time-varying delays were described by a Markov chain. Parameter uncertainties and H_∞ control problem were not considered in [85].

Second, we addressed optimal co-design issues for NCSs. Co-design considerations including network parameters, control parameters, and NCS performances were presented as the design guideline for NCSs. Based on a design chart modified after [38] and a new NCS model proposed in Chapter III, a quantitative method about how to determine the location of the performance degradation points in the NCS performance design chart was presented. The optimal working range of the sampling frequency was also determined based on the locations of these points. To improve the overall QoC of NCSs, we also described how optimal controllers can be designed for QoC optimization.

With the case study of the ball maglev NCS test bed, the feasibility of the optimal co-design methodology proposed in this chapter was verified.

CHAPTER VI

NETWORK SCHEDULING AND ADAPTIVE CONTROL CO-DESIGN

In this chapter, a co-design methodology including network bandwidth scheduling and adaptive control design for NCSs that optimizes overall control performance and reduces network-bandwidth usage is presented.

6.1 Introduction

The key advantage of an NCS is that it provides the flexibility to quickly reconfigure its system architecture and to easily share information with other subsystems. The change of the system configuration might also change the time-delay signature of a networked device, thus change the network QoS. To design an NCS, both its control and communication aspects should be considered because the control performance of the NCS's feedback-control loops is limited by the bandwidth of the communication network. For example, the reduction of the sampling interval improves the control loop's performance [78]. However, a shorter sampling interval requires more network bandwidth to transmit more sensor or control data, which increases the network traffic load. This may affect the system stability and performance of the control loop if the maximum available network bandwidth is exceeded. Therefore, a co-design of control and communication system must be applied in designing an NCS. To explicit this co-design issue, a dynamic optimal network-bandwidth-allocation (ONBA) algorithm based on control system's QoP and an adaptive controller design based on communication system's QoS are proposed in

this chapter. The objective of co-design is to design a networked controller that can adaptively modify the control algorithm according to the control QoP and network QoS.

Traditionally, research on bandwidth allocation and scheduling techniques focused on static strategies that would guarantee average control performance at the expense of permanently occupying the available bandwidth. From the control perspective, the static bandwidth allocation method is an "open-loop" solution because once established at system set-up, the static scheduling will not be adjusted at run time. However, due to network bandwidth limitation in some cases, not all control loops can simultaneously gain enough bandwidth allocation to provide the best possible control performance. Static techniques may not be efficient when changing conditions occurs at the control-application or network levels, because pre-assigned bandwidth resources could be underutilized. Ideally, these underutilized resources could be made available to other applications to provide new functionality.

We followed a "closed-loop" technique and developed a dynamic ONBA algorithm that makes scheduling decisions based on the performance information of each control loop as shown in Fig. 6-1. Bandwidth allocation (BA) is implemented in each control loop, and network bandwidth is dynamically assigned to each control loop according to the control performance (denoted as E_i in Fig. 6-1).

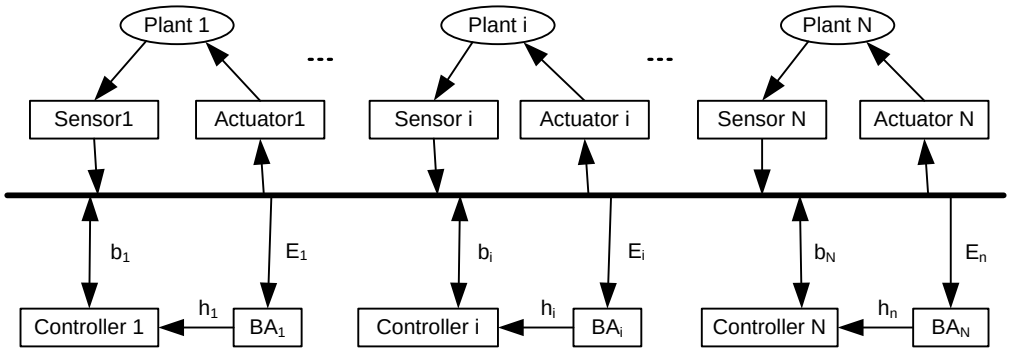


Fig. 6-1. NCSs with ONBA

For each control loop, improved QoC can be achieved if the networked controller can adaptively modify its control algorithm according to the QoS changes like varying time delays and packet losses. This can be formulated as an adaptive control problem which can adjust its parameters on-line according to the changing network QoS parameters. The resulting control algorithm would explicitly depend on the current measurements of network QoS parameters. The two main network-induced effects, time delays and data-packet losses, are summarized by QoS in this chapter. Other QoS-criteria are not considered in this control perspective.

6.2 Dynamic Optimal Bandwidth Allocation

6.2.1 Problem Statement

We consider the NCSs shown in Fig. 6-1 with N control loops, each one controlling a plant. With the consideration of time-varying sampling frequencies, each plant in closed loop can be described by the following system model

$$\mathbf{x}_i(k+1) = \Phi(h_{i,k})\mathbf{x}_i(k) + \Gamma(h_{i,k})\mathbf{u}_i(k) \qquad i = 1, \ ..., \ N \tag{6.1}$$

where $\mathbf{x}_i(k) \in R^n$ is the state of Plant i, $\mathbf{u}_i(k) \in R^m$ is the control input for Control Loop i, $h_{i,k}$ denotes the sampling period of Control Loop i at the time instant k, and $\Phi(h_{i,k})$ and $\Gamma(h_{i,k})$ are real matrix functions of $h_{i,k}$ of appropriate dimensions.

The sampling period $h_{i,k}$ can be obtained from the bandwidth utilization $b_{i,k}$ to be assigned to Control Loop i at time instant k according to the following equation [49].

$$h_{i,k} = \frac{\tau_i}{b_{i,k}}, 0 \le b_{i,k} \le 1, \tag{6.2}$$

where the operation time τ_i denotes the time required to finish a control operation for Control Loop i in the best case, which only includes the time for data processing such as sensor sampling, controller calculating, and actuator actuating, and the time for transmitting the data packets from the sensor node to the controller node and from the controller node to the actuator node. The bandwidth utilization $b_{i,k}$ is the parameter that indicates the portion of the network bandwidth assigned to Control Loop i at time instant k. The network utilization is defined as the ratio of the total time used to transmit data and the total running time, which the sum of the ration of message transmission times and message periods of all devices. If the network utilization approaches zero, there is network bandwidth available for other functionalities or control purposes. If the network utilization approaches one, the network becomes saturated, and it is difficult to increase the sampling rates of control devices or add more devices. Then network bandwidth reallocation to reassign the traffic load or network redesign is needed.

We assume that the periodic sensor data from the sensor node and the control data from the controller node are packetized to an identical bit length L_i. If the data rate of the network medium is R, the best-case one-way data transmission time is

$$T_{i,t} = \frac{L_i}{R}. \tag{6.3}$$

Let $T_{i,p}$ be the time needed for data processing such as sensor sampling (A/D conversion), actuator actuating (D/A conversion), and controller calculating the control data for Control Loop i in the best case. The operation time τ_i can be expressed as

$$\tau_i = 2T_{i,t} + T_{i,p} . \tag{6.4}$$

For each known Control Loop i, $T_{i,p}$ and $T_{i,t}$ in the best case can be measured and computed. Thus from (6.4) we can assume that τ_i is a constant, then smaller $h_{i,k}$ indicates bigger $b_{i,k}$ from (6.2). The rationale behind (6.2) is that a higher sampling frequency requires more bandwidth allocation to transmit more data. Related to (6.2), there are several special cases.

1. When $h_{i,k} = \tau_i$, i.e., $b_{i,k} = 1$, the 100% of the network bandwidth is used by Control Loop i, and no other control loops are allowed to share the network medium. This is the case of NCSs with only one control loop.
2. When $h_{i,k} = D_i$, where D_i is the maximum allowable loop delay (MALD) for Control Loop i, the minimum network bandwidth utilization of Control Loop i is

$$(b_{i,k})_{\min} = \tau_i / D_i . \tag{6.5}$$

3. When there are N control loops, the most available bandwidth could be assigned to Control Loop i is (6.6) while all the other control loops are assumed to use their minimum bandwidth, i.e.

$$(b_{i,k})_{\max} = 1 - \sum_{j \neq i}^{N} (b_{j,k})_{\min} . \tag{6.6}$$

In some cases, due to network bandwidth limitation, not all systems can simultaneously obtain enough bandwidth allocation to transfer data and execute at their highest sampling frequency. Investigating how to perform the optimal bandwidth allocation to obtain the optimal

control performance for each closed-loop system thus to optimize the overall system QoC is the main objective of this chapter. The following rationale is considered: when a controlled plant is in equilibrium, the assigned execution rate (or sampling period) may not be required. That is, the assigned bandwidth may be wasted, and it can be reduced for the sake of saving bandwidth usage and enhancing the bandwidth utilization of other control loops. On the other hand, when a controlled plant is affected by a perturbation, increasing its assigned bandwidth by adding the underutilized bandwidths of other control loops in equilibrium may hasten system recovery from the perturbation and improve its system performance. This network-bandwidth reallocation is particularly useful when the overall network bandwidth resource is limited.

Ray and Halevi [25] showed that the feedback-control performance directly depends on the loop delay, which is defined as the interval between the instant when the sensor node samples data and the instant when the actuator actuates the control command. The MALD of Control Loop *i* can be obtained from a conventional stability criterion and performance analysis. In order to guarantee system stability and control performance, two control measures can be used to determine the maximum allowable loop delay [78]: phase margin ϕ and the closed-loop bandwidth ω_{bw}. To guarantee acceptable control performance such as response rapidness and smoothness, the "rule of thumb" for selecting the sampling frequencies mentioned in Section 5.3.2 can be used to estimate the MALD D_i for Control Loop *i*. From (5.53–5.54), D_i can be estimated by

$$D_i \le T_{i,bw} / 20 , \tag{6.7}$$

where $T_{i,bw} = 2\pi / \omega_{i,bw}$ and $\omega_{i,bw}$ is the closed-loop bandwidth of Control Loop *i*.

Alternatively, from (5.55) the MALD D_i could also be estimated by (6.8)

$$D_i \le t_{i,r} / 4, \tag{6.8}$$

where $t_{i,r}$ is the rise time of closed-loop system i.

Let $e_{i,k}$ denote the error of Control Loop i, the distance of the state from its equilibrium point at time instant k. It can be expressed in regulation problems [55–56] as following

$$e_{i,k} = \left\| \mathbf{x}_i(k) \right\|. \tag{6.9}$$

For Control Loop i, we can define a performance criterion that relates control performance (such as the error) with bandwidth utilization as

$$e_{i,k} = E(b_{i,k}). \tag{6.10}$$

In general, the less the bandwidth allocation is, the worse the control performance (i.e., the larger the error). Thus (6.10) can be approximated by a linear relation as [49]:

$$e_{i,k} = E(b_{i,k}) \approx \frac{\beta_i}{b_{i,k}}, \tag{6.11}$$

where the parameter β_i is specific to each control loop and can be determined prior to the implementation of the NCSs by evaluating the control performance of each control loop for a broad range of sampling rates or bandwidth allocations. Such linearization method was mentioned in [56].

6.2.2 Dynamic ONBA Algorithm

The dynamic ONBA problem is how to dynamically assign a bandwidth utilization b_i to each control loop according to the control performance and network bandwidth availability such that the overall QoP of the NCSs is optimized. The development of this dynamic ONBA algorithm is presented in this section.

The constraint of bandwidth allocation is $\sum_{i=1}^{N} b_{i,k} \le 1$, i.e., the total bandwidth utilization must not exceed the whole network capacity. Then the current additionally available bandwidth utilization is

$$b_a = 1 - \sum_{i=1}^{N} b_{i,k} \,. \tag{6.12}$$

If $\sum_{i=1}^{N} (b_{i,k})_{\min} \ge 1$, then the NCSs are not schedulable with the current choice of network medium. Choosing another network or reducing the number of the control loops is needed.

If each control loop is allocated with a fixed bandwidth, there may be a waste of the network bandwidth since each control loop has its own control and traffic requirement and some control loops may not necessarily need a fixed bandwidth when they are in equilibrium. In order to provide services to the maximal number of control loops with their QoC requirements and to achieve high utilization of the bandwidth resources, the bandwidth allocated to each control loop needs to be minimized.

Thus we formulate the following cost function to be minimized

$$J_{i,k} = a_{i,1} {e_{i,k}}^2 + a_{i,2} {b_{i,k}}^2 , \tag{6.13}$$

where $a_{i,1}$ and $a_{i,2}$ are the weighting coefficients for Control Loop i. The optimization object of bandwidth allocation is to find a suitable bandwidth utilization $b_{i,k}$ that can minimize the network bandwidth usage and maximize the system performance (i.e. minimize the error) as well.

Considering all the control loops, the optimization function J for the whole system becomes the following:

$$J_k = \sum_{i=1}^{N} J_{i,k} = \sum_{i=1}^{N} (a_{i,1} {e_{i,k}}^2 + a_{i,2} {b_{i,k}}^2) . \tag{6.14}$$

Since all the N control loops are assumed to be independent, if each $J_{i,k}$ of the ith control loop is minimal, then J_k is also the minimal. Hence, the optimal bandwidth allocation for each control loop can achieve the QoP optimization of the overall system with the cost function defined in (6.14).

There can be three notable special cases:

1. When Control Loop i is in equilibrium, i.e. $e_{i,k} = 0$, from (6.5) and (6.13) we have the optimal bandwidth allocation for Control Loop i as

$$(b_{i,k})_{\text{opt}} = (b_{i,k})_{\min} = \tau_i / D_i , \tag{6.15}$$

where $(b_{i,k})_{\text{opt}}$ is the optimal bandwidth utilization for Control Loop i at time instant k. Substituting (6.7) or (6.8) into (6.15) obtain following (6.16) or (6.17).

$$(b_{i,k})_{\text{opt}} = 20\, \tau_i / T_{i,bw} , \tag{6.16}$$

$$(b_{i,k})_{\text{opt}} = 4\, \tau_i / t_{i,r} . \tag{6.17}$$

With (6.2), the corresponding optimal sampling period is obtained as

$$(h_{i,k})_{\text{opt}} = T_{i,bw} / 20, \tag{6.18}$$

or

$$(h_{i,k})_{\text{opt}} = t_{i,r} / 4, \tag{6.19}$$

2. When Control Loop i experiences perturbation, $e_{i,k} \neq 0$. Substituting (6.11) into (6.13) and differentiating $J_{i,k}$ with respect to $b_{i,k}$, the optimal value $(b_{i,k})_{\text{opt}}$ can be obtained as

$$(b_{i,k})_{\text{opt}} = \sqrt[4]{\frac{a_{i,1}\beta_i^3}{a_{i,2}}}. \tag{6.20}$$

With (6.2), the corresponding optimal sampling period is

$$(h_{i,k})_{\text{opt}} = \frac{\tau_i}{\sqrt[4]{\frac{a_{i,1}\beta_i^3}{a_{i,2}}}}. \tag{6.21}$$

3. For Control Loop i with the highest processing demands, i.e., $h_{i,k} \leq h_{j,k}, j = 1,..., N, j \neq i,$ all the additional available bandwidth allocation can be assigned to further improve its control performance, i.e. $(b_{i,k})_{\text{opt}} = \sqrt[4]{\frac{a_{i,1}\beta_i^3}{a_{i,2}}} + b_a$, where b_a is the current additionally available bandwidth utilization defined in (6.12). Thus

$$(b_{i,k})_{\text{opt}} = \sqrt[4]{\frac{a_1\beta_i^3}{a_2}} + 1 - \sum_{i}^{N} b_{i,k} = 1 - \sum_{j \neq i}^{N} b_{j,k}, \tag{6.22}$$

and the optimal sampling period is

$$(h_{i,k})_{\text{opt}} = \frac{\tau_i}{1 - \sum_{j \neq i}^{N} b_{j,k}}. \tag{6.23}$$

Based on the analysis above, the ONBA algorithm can be summarized in Fig. 6-2.

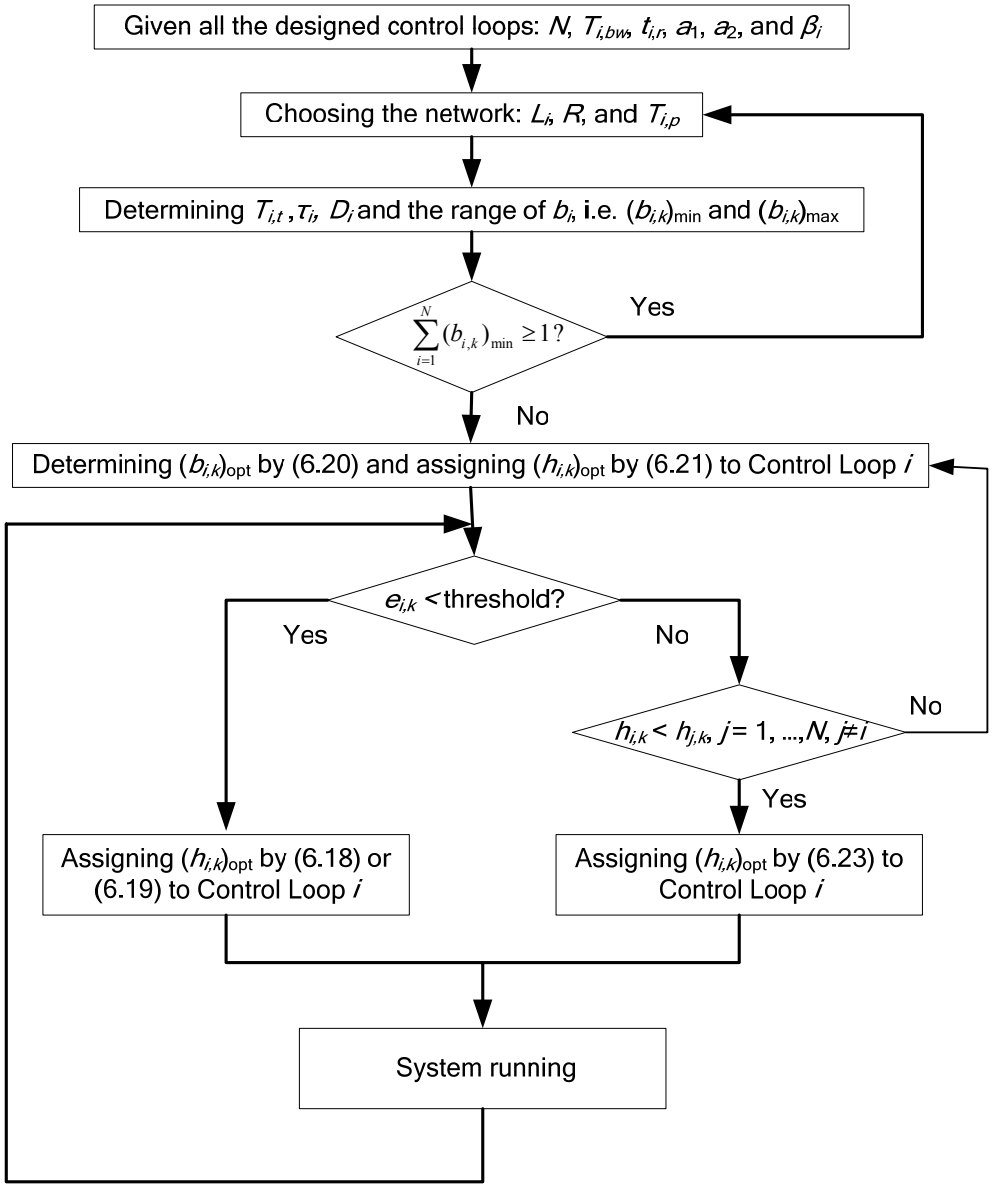


Fig. 6-2. Dynamic ONBA algorithm

The ONBA algorithm presented in Fig. 6-2 can be implemented as a part of the control algorithm in each control loop. For each control loop, the controllers with three different sampling periods are designed prior to the system implementation. During the system run-time, each control loop keeps monitoring the system error to check if it is zero (or within a preset threshold), then the decision of which controller should be used is made based on this system error information. If system error is zero (or within a preset threshold), then the smallest network bandwidth utilization (largest sampling period) is assigned to this control loop. When the system error is large (experiencing perturbation), there are two cases. If the current control loop has the highest processing demands, then the optimal bandwidth utilization based on (6.22) is assigned. i.e., all the currently available bandwidth can be assigned to this control loop to ensure its best QoP. Otherwise, the optimal bandwidth utilization based on (6.20) is assigned to optimize the overall QoP.

Remark 6-1: This technique requires controllers capable of running with different sampling frequencies. For systems given by (6.1), controllers are designed specifying of three sampling periods using (6.18) or (6.19), (6.21), and (6.23), for which the closed-loop stability and performance requirements are met, adapting the gains accordingly. These three controllers can be designed prior to the system run-time. Real-time implementation of this ONBA algorithm is feasible because there is no computational complexity.

Remark 6-2: This algorithm can be easily extended to cover the case that there are two or more control loops simultaneously experience perturbations by introducing prioritization mechanism. Each control loop can be assigned a priority number according to their processing demands and QoP specifications. The decision of assigning additional bandwidth can be made based on the priorities of these control loops.

6.2.3 Simulation Verification

In order to verify the ONBA algorithm developed above, some simulations developed using MATLAB Simulink are presented. The block diagram of this simulation is presented in Appendix B.4.

The illustrative NCS simulation contains 5 independent control loops similar with the configuration shown in Fig. 6-1, each consisting of a sensor, a lead controller, an actuator, and a DC motor as the controlled plant. The system model of the DC motor is given as [78]

$$G(s) = \frac{1}{s(s+1)}. \tag{6.24}$$

The lead controller is given as [78]

$$D(s) = 10\frac{s/2+1}{s/10+1}. \tag{6.25}$$

All controllers and all DC motors have been defined to be the same because it simplifies the performance analysis and comparison. We assume the network and control parameters as $\tau = 0.03\,\text{s}, D = 0.3\,\text{s},$ where τ is the time required to finish a closed-loop control operation and D is the maximum loop delay as defined in (6.5). From (6.5–6.6), we can obtain the working range of bandwidth allocation as

$$\begin{aligned} (b_k)_{\min} &= \tau / D = 0.1, \\ (b_k)_{\max} &= 1 - 4 \times (b_k)_{\min} = 0.6. \end{aligned} \tag{6.26}$$

The corresponding sampling periods are obtained as $h_{\min} = 0.05\,\text{s}, h_{\max} = 0.3\,\text{s}$ from (6.2).

For a static strategy, we assume all the plants share the bandwidth equally and there is no other traffic load, thus the bandwidth allocation for each control loop by using static strategy is $b_k = 1/5 = 0.2$, and the sampling period for each DC motor is $h_k = \tau / b_k = 0.03/0.2 = 0.15\,\text{s}$.

A static bandwidth allocation strategy and the ONBA strategy were both implemented to provide a direct comparison between their performances. In the simulation, 5 periodic step disturbances with different phase delays were inputted to these 5 DC motors, respectively. The system responses of these 5 DC motors are identical except the time shift, and the system response of one of the DC motors with two strategies are shown in Fig. 6-3. As evidenced from Fig. 6-3, the performance of system with ONBA (in solid line) is better than that with static strategy (in dotted line).

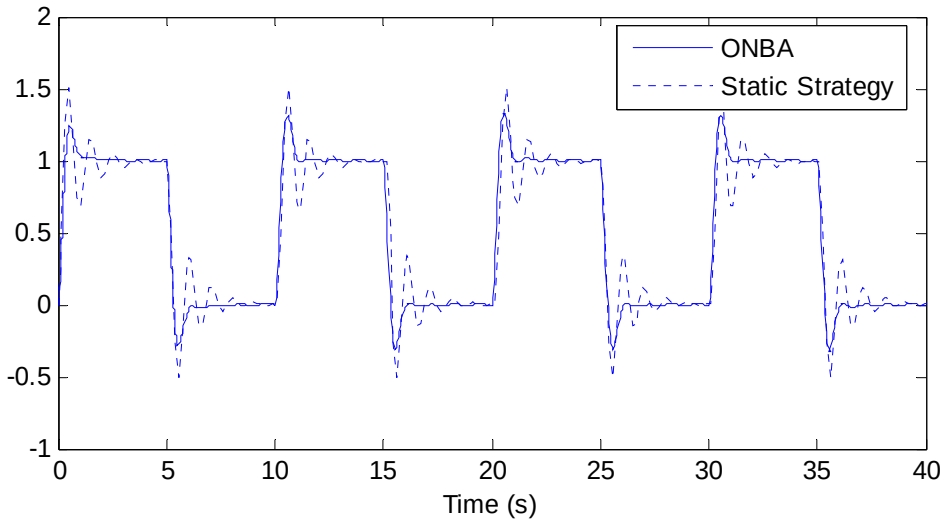


Fig. 6-3. Comparison of system responses of periodic step disturbances

Fig. 6-4 shows the comparison of the cumulative system errors from these two strategies. The closed-loop system error is defined as the absolute difference between the desired response

(set point) and the actual response (feedback output) of the controlled plant. The cumulative system error E is the total cumulative closed-loop system error of all the 5 DC motor control loops, i.e. $E(t) = \int_0^t \sum_{i=1}^{5} |e_i(s)| ds$. As evidenced from Fig. 6-4, ONBA (in dash line) achieves better performance than the static strategy (in dotted line) by reducing about 50% of the cumulative system error.

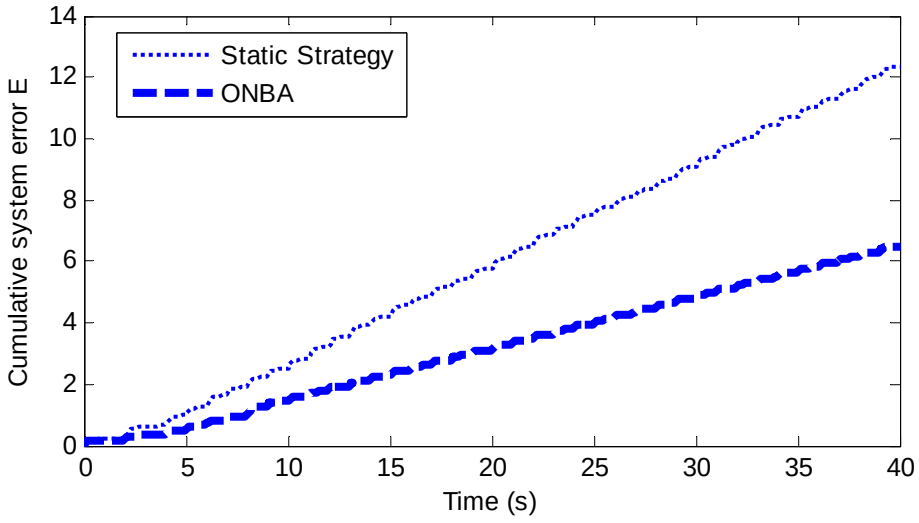


Fig. 6-4. Cumulative system errors comparison

Fig. 6-5 shows the comparison of total bandwidth usages between two systems with these two strategies. Straight line in Fig. 6-5 (a) shows that in static strategy, network bandwidth is totally occupied by the 5 DC motor control loops, no more plants or functionalities could be added in this system. In Fig. 6-5 (b), the network bandwidth usage goes up to 100% only when there are disturbances in the control loops and most of the time the bandwidth occupancy is 50%. Thus some network resource is saved and more control loops or functionalities could be added in

this system. From Figs. 6-3–6-5, it can be concluded that the ONBA algorithm achieves better control performance while uses less network bandwidth than the static strategy.

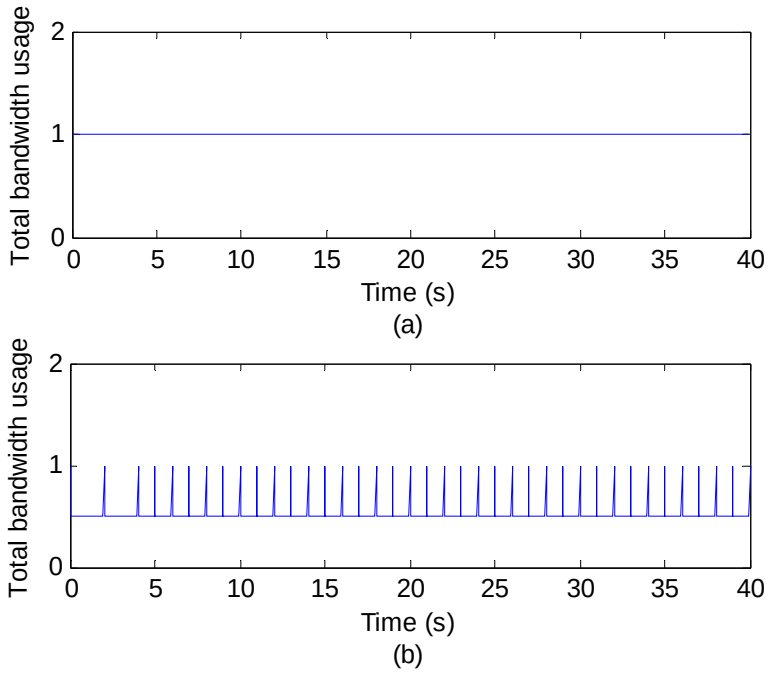


Fig. 6-5. Bandwidth usages of (a) a system with a static strategy (b) a system with ONBA

6.3 Adaptive Control Design

6.3.1 Problem Statement

We consider a control loop in an NCS shown in Fig. 6-6. The sensor sampling period is h, i_k denotes the index of sensor sampling instant, and i_m denotes the index of controller calculation. $r(i_m)$ denotes the input. The following assumptions are made in this section.

6-1. Time delays and packet losses are bounded and only exist between the sensor node and the controller node. The delay range is between τ_{min} and τ_{max}.

6-2. The sensor is clock-driven while the controller and actuator are event-driven. The controller only performs a new calculation after a sensor-data-packet has been received.

6-3. For each sensor-data-packet, the time delay τ_{sc} can be measured by assigning each packet a timestamp. With this timestamp, each data-packet is also numbered.

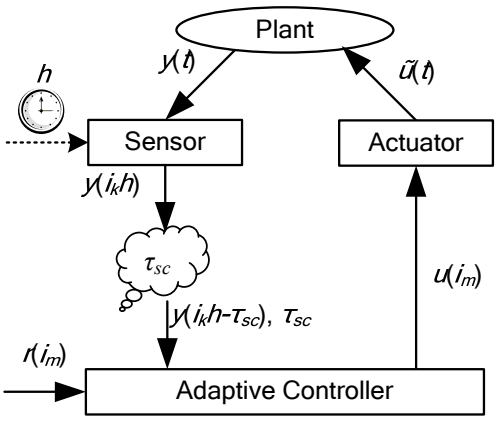


Fig. 6-6. NCS architecture with one-way time delay

To evaluate the QoC, the following measure is defined as [86]:

$$QoC = \frac{1}{IAE}, \tag{6.27}$$

where IAE is the control performance measure defined in (5.56). Equation (6.27) basically indicates that lower integrated absolute error means better quality of control. A QoS-adaptive controller design method to increase the QoC defined above is proposed in detail in the following section.

6.3.2 Adaptive Controller Design

For the plant shown in Fig. 6-6, a continuous-time controller with n control parameters can be designed by standard methodologies before introducing network. The discrete-time form of the controller can be derived using continuous-time to discrete-time transformation, for example with Tustin's method. The adaptive control design procedure can be summarized as the following steps.

1. A simulation is conducted to search the n optimal control parameters that maximize the QoC to be the original set of control parameters.
2. In this step, l values of the time delay τ_l distributed over the range between τ_{min} and τ_{max} are used as constant delay in simulation to search the optimal control parameters according to different delays. For each τ_l, n optimal control parameters that maximize the QoC can be found by simulation. Then l sets of control parameters can be found and defined as

$$K_{\tau_l} = (k_1,...,k_{n-1},k_n)_l,\ l = 1, 2, 3, \ldots . \tag{6.28}$$

   Each K_{τ_l} represents the optimal set of control parameters with n components in terms of QoC according to a specific constant delay τ_l. This l sets of parameters are then stored in a look-up table in the controller node.
3. In this step, network is introduced between the sensor node and the controller node. At each time instant when the sensor-data packet arrives at the controller node, the time delay τ_{sc} is measured by the controller by checking the time stamp. Based on this real-time measured time delay, appropriate set of control parameters are selected from K_{τ_l} in (6.28) which were stored in a look-up table are used in current control algorithm. Since there are only l sets of control parameters obtained from the second step, the set of parameters according to the

delay that is closest to the measured delay will be used. If there is no delay, the original set of parameters obtained from the first step will be used.

To deal with the packet losses, varying sampling periods can be used according to the data-packet-loss rate. Based on Assumption 3, each data-packet is numbered, let i_k be the packet number, then the varying sampling period used in control algorithm is given by

$$h_{i_m} = h\,(i_k - i_{k-1}),\ i_k \in \{1, 2, 3, ...\}\,, \tag{6.29}$$

where h_{i_m} is the sampling period used in the i_m th calculation of the controller, i_k and $i_{k\text{-}1}$ are the packet numbers of the current and the last received packet, respectively. If there is no packet loss, from Remark 3-1, then $i_k = i_{k-1} + 1$, thus h_{i_m} is equal to h.

Remark 6-3: The adaptive control method can be extended to cover the situation when time delays and packet losses both occur during the data communication process. This can be done by introducing one more valuable, sampling period, to the aforementioned look-up table. i.e., the optimal set of control parameters K_{τ_l} is evaluated not only based on the time delay, but also the sampling period. Thus during the run-time, before a new control calculation is performed, the optimal set of control parameters K_{τ_l}, and the new sampling period h_{i_m} must be chosen.

Remark 6-4: The QoS-adaptive controller design approach developed in this section has the similar design mechanism with the ONBA algorithm proposed in previous section. They both follow the "real-time feedback" technique to improve the controller design. The ONBA algorithm in network scheduling problem searches optimal sampling period for controller design

based on control system QoP, while the QoS-adaptive control approach searches optimal control parameter for controller design based on communication system QoS. The internal connection between these two problems verifies that NCS design is a multidisciplinary field that includes the co-design of control systems and communication systems. The network QoS should be analyzed together with specifying the control QoP before implementing the real-time control over networks.

6.3.3 Simulation Example

A simulation example is presented in this section to demonstrate how the adaptive controller design proposed above can be performed. This example is the DC motor example taken out of [7]. The plant is given by

$$G(s) = \frac{2029.826}{(s + 26.29)(s + 2.296)}. \tag{6.30}$$

The PI controller that has been designed is given by [7]

$$D(s) = \frac{\beta k_P (s + (k_I / k_P))}{s}. \tag{6.31}$$

where k_P = 0.1701, k_I =0.378, and β is a parameter to adjust controller gains. How to choose β to obtain optimal control performance when there is time delay in the control loop is investigated in [7]. If there is no time delay, β = 1. Using MATLAB "c2d" command based on Tustin's method, the following discrete-time control algorithm can be obtained.

$$u(i_k h) = u(i_{k-1} h) + [k_p + 0.5 k_I h] \cdot e(i_k h) + [0.5 k_I h - k_p] \cdot e(i_{k-1} h) . \quad (6.32)$$

Thus in this example, the number of the control parameters is 2. i.e., $n = 2$, $k_1 = \beta k_P$, and $k_2 = \beta k_I$. The sampling period was set to be 30 ms representing 25% of the rising time of the continuous closed loop based on the "rule of thumb" in (6.19). i.e., $h = 30$ ms.

Several simulations were conducted to search the optimal sets of control parameters (k_1, k_2) for different time delays. Table 6-1 shows the resulted look-up table that relates the optimal sets of control parameters and different time delays.

Table 6-1. Optimal control parameters with different time delays ($h = 30$ ms)

Time Delay (ms)	k_1	k_2	β
0	0.1701	0.3780	1
15	0.1531	0.3402	0.9
30	0.1361	0.3024	0.8
45	0.1106	0.2457	0.65
60	0.0851	0.1890	0.5
75	0.0680	0.1512	0.4
90	0.0501	0.1134	0.3

Then random delays were introduced between the controller node and the sensor node in the DC motor system. The time delays were varying between 10 ms and 100 ms. The block diagram of this simulation is presented in Appendix B.5. PI controllers with control parameters in Table 6-1 were implemented respectively. QoS-adaptive controller proposed in previous section was also implemented and it used the look-up table obtained above. Fig. 6-7 shows a comparison of system performance between the adaptive controller and a non-adaptive controller. A unit pulse with a period of 4 ms is used as the input. The simulation period is 12 s. The solid line in Fig. 6-7 denotes the system response with adaptive controller and the dotted line denotes the system response with non-adaptive controller with control parameters (k_1 = 0.1361, k_2

=0.3024). As evidenced from Fig. 6-7, the adaptive controller achieves better control performance than non-adaptive controller.

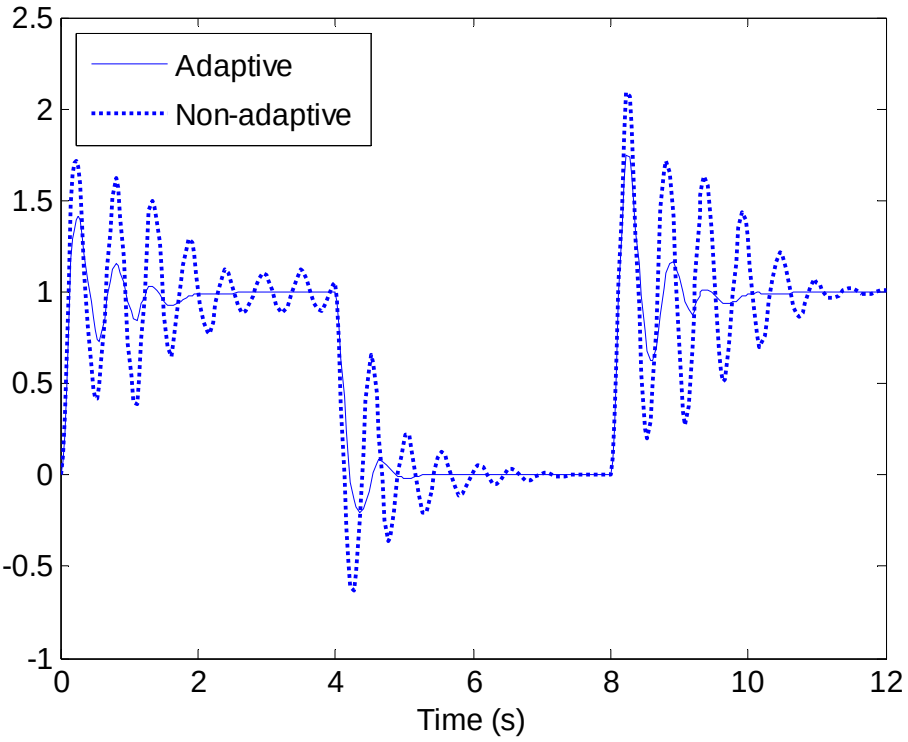


Fig. 6-7. System responses with adaptive controller and non-adaptive controller (k_1 = 0.1361 and k_2 = 0.3024)

The QoC measure defined in (6.27) was used to further verify the effectiveness of the proposed adaptive controller. Simulations of the DC motor system with different sets of control parameters shown in Table 6-1 were conducted and corresponding QoC values are calculated and shown in Table 6-2. The QoC value of system with adaptive controller is also shown in this table to have a direct comparison with these non-adaptive controllers. Table 6-2 proves that QoC improvement can be achieved by implementing the QoS-adaptive controller proposed in this chapter.

Table 6-2. QoC values with different control parameters

QoC	Non-adaptive Controller		
	k_1	k_2	β
0.352	0.1701	0.3780	1
0.5263	0.1531	0.3402	0.9
0.5556	0.1361	0.3024	0.8
0.5405	0.1106	0.2457	0.65
0.5882	0.0851	0.1890	0.5
0.625	0.0680	0.1512	0.4
0.5714	0.0501	0.1134	0.3
0.9091	Adaptive Controller		

6.4 Summary

In this chapter, the co-design of network bandwidth allocation and adaptive control was proposed. As a part of this co-design methodology, a "closed-loop" optimal network bandwidth allocation algorithm for NCSs with communication constraints was presented. The proposed dynamic strategy integrates feedback control with real-time scheduling. The ONBA algorithm makes scheduling decisions based on the dynamic QoP information of each control loop. With this algorithm, there is less underutilized bandwidth resource, and the QoP optimization of the overall NCSs can be achieved. The simulation results showed that this optimal bandwidth allocation approach better utilizes the bandwidth resource under the constraint of the limited available bandwidth in comparison with the static bandwidth allocation approach.

As another part of the co-design methodology, a QoS-adaptive control design approach was also presented. The idea is based on calculating new control values with reference to the QoS parameter such as time delays and packet losses measured online. With the assumption that the time delays and packet losses only exist between the sensor node and the controller node, the simulation results showed that control performance can be improved by implementing the QoS-adaptive controller proposed in this chapter.

CHAPTER VII

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

This chapter describes the key achievements in this research and suggestions for future work.

7.1 Conclusions

With the advancement in the automation industry, the need to perform complex remote operations has grown significantly. Ever-increasing computational capabilities and advancements in the networking technology enabled researchers to develop NCSs to implement the control scheme of real-time control over networks. However, the introduction of network communications in the closed-loop system's dynamics would inevitably produce additional uncertainties of time-delay and data-packet losses. They would come from the nondeterministic factors during the physical signal coding and processing and the data transmission process. Time sharing of the communication medium in an NCS also introduces a serious network scheduling problem since the network bandwidth is usually limited. These uncertainties could degrade the system performance and even cause system instability and should be analyzed and dealt with. Control design methodologies to further improve the control performance should also be investigated. To cover these issues in real-time control over networks, this dissertation presented a framework for the modeling, analysis, delay/data-loss compensation, advanced control, optimal network scheduling, and experimental verification of NCSs. These results are summarized in the

following paragraphs.

Based on the ground work performed by Srivastava and Ambike [21–22], the feasibility of Internet-based real-time control of a ball maglev setup was experimentally verified in this dissertation. Internet-based supervisory control and feedback control over the Ethernet were presented. The real-time operating environment was proposed as a solution to properly time key communication events in an NCS and to have complete control on their execution, reducing the device processing time. Feedback control over the Ethernet based on Linux with RTAI was demonstrated as one of the solutions of real-time control over networks by the successful implementation of the real-time control of the ball maglev setup over an Ethernet.

This dissertation analyzed networked feedback control architecture and formulated a delay-dependent dynamic model in which the effects of time-delay, data-packet-loss, and out-of-order data transmission were all considered. Based on this model, a new delay-dependent stability criterion and the upper bound of the time delays that the system could accommodate were derived through a Lyapunov functional approach. The appropriate co-design integration of control systems, real-time systems, and network communication systems proposed in this dissertation is based on this delay-dependent dynamic model. The detailed explanation of this co-design consideration is presented in the following paragraph.

In order to improve the performance of the real-time control over networks, as shown by the solid curve in Fig. 7-1, and to make it close to the performance of the network-free digital control system, as shown by the dashed curve in Fig. 7-1, a systematic co-design consideration that utilizing both network and control parameters was proposed in this dissertation. This systematic consideration includes delay/data-loss compensation, robust control design for uncertainties, optimal control design to identify the best working range of sampling frequencies, and optimal network bandwidth allocation. All of these issues were addressed in this dissertation

to improve the NCS performance, i.e., increase the shadowed area shown in Fig. 7-1.

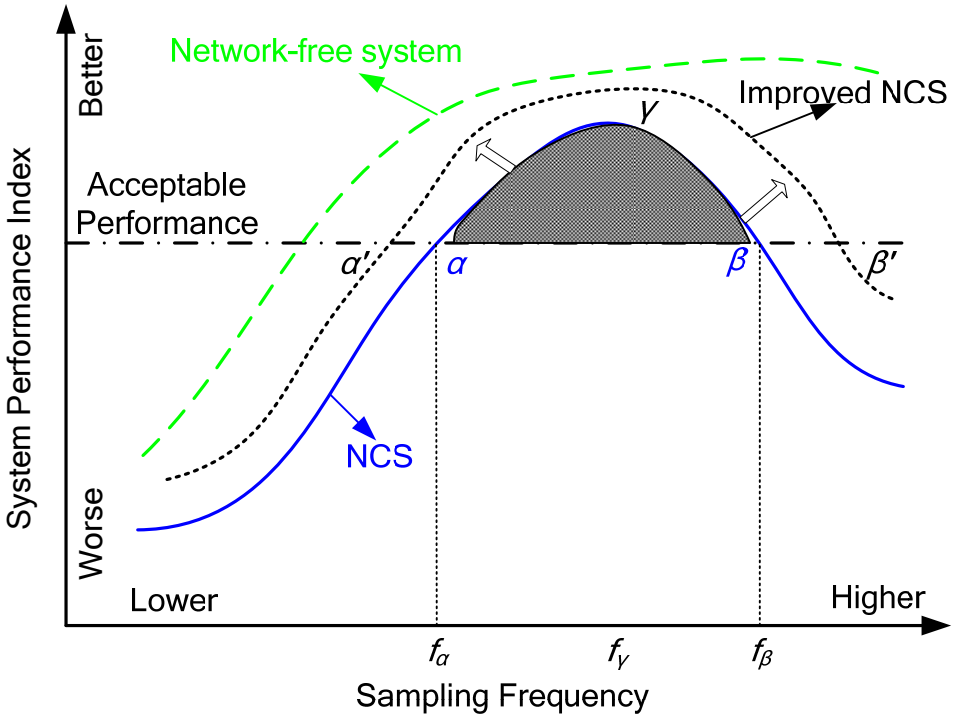


Fig. 7-1. NCS performance chart. Modified after [38]

To ensure the system stability of the ball maglev setup in the presence of network-induced sporadic delays and successive data-packet losses, two new compensation algorithms based on model-estimation and sensor-date prediction were proposed. An augmented system model with the model-estimation-based algorithm was analyzed, and the stability analysis of the compensated system was presented. Experiment results verified the effectiveness of these two compensation algorithms. These delay/data-loss compensation algorithms enable the NCS to use a higher sampling frequency to achieve better system performance.

After the successful implementation of real-time control over the Ethernet, advanced

control design methodologies were investigated to overcome the uncertainties and further improve the system performance. Robust control problems for NCSs with time-delay and parameter uncertainties were formulated. Robust stabilization, robust H_∞ control, and robust-control-parameter-optimization problems were addressed. Delay-dependent methods of designing linear memoryless state-feedback controllers and dynamic state-feedback controllers to solve robust control problems were presented. The MATLAB simulation and the experiment setup of the ball maglev system were both used as a verification tool for the robust control methodology.

The NCS performance chart shown in Fig. 7-1 reveals the existence and locations of the performance degradation points that determine the best working range of sampling frequencies. Based on the aforementioned new NCS model, a quantitative method about how to estimate this working range was presented in detail. With this sampling-frequency range, an optimal controller design method was also presented as an important part of the optimal co-design methodology for NCSs. The optimal co-design procedures based on Fig. 7-1 can be summarized as follows [38]. (1) If the actual network is unavailable at the first stage of the design process, we can utilize the control performance analysis and the stability criteria developed in Chapter III to investigate the feasibility of the chosen system parameters. (2) If a candidate network is available at the next design stage, actual information on device processing times and network traffic can be collected. During this design stage, an NCS performance design chart as shown in Fig. 7-1 can be drawn, and Points α, β and γ can be determined. (3) After the working range of sampling frequencies for stability is derived by the method developed in Chapter V, different combinations of network parameters can be studied to optimize the control QoP that depends on other design objectives. (4) In order to guarantee the best QoC of NCSs, various controller designs with different sampling frequencies from this working range can also be tested to verify their stability and

performance. A case study of the ball maglev system demonstrated the feasibility of the proposed optimal co-design methodology.

As pointed out in Fig. 7-1, the performance of an NCS is affected by the sampling frequency and the network-traffic load, especially when there are many control loops sharing the network medium and competing against each other for the limited network bandwidth. Increasing the sampling frequency may improve the control loop's performance. However, a higher sampling frequency requires more network bandwidth to transmit more sensor or control data, which increases the network-traffic load. This might affect the entire system stability and the performances of some other control loops if the maximum available network bandwidth should be exceeded. To resolve this problem, a "closed-loop" optimal network-bandwidth-allocation algorithm for NCSs with communication constraint was presented. The proposed dynamic strategy integrates feedback control with real-time network scheduling. The dynamic ONBA algorithm makes scheduling decisions based on the real-time performance of each control loop. The simulation results showed that the optimal bandwidth allocation approach better utilized the bandwidth and could support more control loops under the constraint of the limited bandwidth in comparison with the static-bandwidth-allocation approach.

Followed the design mechanism behind the ONBA algorithm, an adaptive controller design approach was presented as a part of the co-deign methodology. The proposed adaptive controller modifies its control algorithm based on the changing QoS parameters such as time delays and packet losses that are measured online.

7.2 Suggestions for Future Work

This dissertation provides the foundation for future research efforts in real-time NCS

design and analysis. In this section, a few possible further research directions are explored in detail.

For network devices, the device processing time is a significant factor that affects the total time delay. This dissertation verified using real-time operating environment to improve the device processing capability. The device processing capability also relies on the communication protocol. Hence, in order to improve the utilization of network bandwidth and enhance the networked device processing capability, deterministic protocols specially designed for NCS purpose should be further developed.

To guarantee the stability and performance of the overall NCS, including network systems and control systems, a complete analysis and design solution of hierarchical control systems that include different types of system models such as continuous plants and discrete-time controllers and real-time or non-real-time multivariable data transmissions should be further investigated.

Optimal scheduling also deserves more efforts. Different performance measure functions should be proposed and analyzed for effectiveness. It would be interesting to explore the scheduling problem of NCSs with known approaches proposed in other scheduling problems in such as broadcast disk systems [87] and multimedia transmissions [88].

The adaptive controller design proposed in Chapter VI deals with only the case of one-way time delay. More work need to be done to deal with the extension of the approach on the general case adding the path from the controller node to the actuator node. The extended case should also include the situation of out-of-order data transmission. Strict optimization attempts might need to be investigated.

REFERENCES

[1] R. C. Geortz, "Manipulator systems developed at ANL," in *Proc. of the 12th Conference on Remote Systems Technology*, American Nuclear Society, vol. 1, pp. 117–136, November 1964.

[2] T. B. Sheridan, "Telerobotics," *Automatica*, vol. 25, no. 4, pp. 487–507, July 1989.

[3] Y. Yokokohji and T. Yoshikawa, "Bilateral control of master-slave manipulators for ideal kinesthetic coupling - formulation and experiment," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 605–620, October 1994.

[4] A. J. Madhani, G. Niemeyer, and J. K. Salisbury Jr., "The black falcon: A teleoperated surgical instrument for minimally invasive surgery," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, B.C., Canada, vol. 2, pp. 936–944, October 1998.

[5] M. Mitsuishi, S. Tomisaki, and T. Yoshidome, "Tele-micro-surgery system with intelligent user interface," in *Proc. of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, vol. 2, pp. 1607–1614, April 2000.

[6] J. H. Park and T. B. Sheridan, "Supervisory teleoperation control using computer graphics," in *Proc. of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, vol. 1, pp. 493–498, April 1991.

[7] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control Engineering Practice*, vol. 11, no. 10, pp. 1099–1111, February 2003.

[8] K. Ji, W. -J. Kim, and A. Srivastava, "Internet-based real-time control architectures with time-delay/packet-loss compensation," *Asian Journal of Control*, vol. 9, no.1, in press, March 2006.

[9] J. Eker and A. Cervin, "Distributed wireless control using Bluetooth," in *Proc. of IFAC*

Conference on New Technologies for Computer Control, Hong Kong, P.R. China, pp. 1–6, November 2001.

[10] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, "Closed-loop control over wireless network," *IEEE Control System Magazine*, vol. 24, no. 3, pp. 58–71, June 2004.

[11] S. Lankes, M. Reke, and A. Jabs, "A time-triggered Ethernet protocol for real-time CORBA" in *Proc. of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, Washington DC, USA, pp. 215–222, April 2002.

[12] C. Venkatramani, *Design, Implementation and Evaluation of RETHER: A Real-Time Ethernet Protocol*, Ph.D. Dissertation, State University of New York at Stony Brook, December 1996.

[13] R. Safaric, M. Debevc, R. M. Parkin, and S. Uran, "Telerobotics experiments via Internet," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 2, pp. 424–431, April 2001.

[14] H. Hu, L. Yu, P. W. Tsui, and Q. Zhou, "Internet-based robotic systems for teleoperation," *International Journal of Assembly Automation*, vol. 21, no. 2, pp. 143–151, May 2001.

[15] T. Sato, J. Ichilawa, M. Mitsuishi, H. Miyazakik, and Y. Hatamura, "Micro-teleoperation with manual task execution posture," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 22–29, February 1995.

[16] C. Smith and P. K. Wright, "Cybercut: A World Wide Web design-to-fabrication tool," *Journal of Manufacturing Systems*, vol. 15, no. 6, pp. 432–442, November 1996.

[17] T. T. Ho and H. Zhang, "Internet-based tele-manipulation," in *Proc. of the IEEE Canadian Conference on Electrical and Computer Engineering*, Edmonton, Alberta, Canada, vol. 3, pp. 1425–1430, May 1999.

[18] K. Goldberg, S. Gentner, C. Sutter, and J. Wiegley, "The Mercury Project: A Feasibility Study for Internet Robots," *IEEE Robotics & Automation Magazine*, vol. 7, no. 1, pp. 35–39, March 2000.

[19] R. C. Luo, J. H. Tzou, and Y. C. Chang, "Desktop rapid prototyping system with supervisory control and monitoring through Internet," *IEEE/ASME Transactions on Mechatronics*, vol. 6, no. 4, pp. 399–409, December 2001.

[20] C. E. Garcia, R. Carelli, J. F. Postigo, and C. Soria, "Supervisory control of a telerobotic system: A hybrid control approach," *Control Engineering Practice*, vol. 11, no. 7, pp. 805–817, July 2003.

[21] A. Srivastava, *Distributed Real-Time Control via the Internet*, M.S. Thesis, Texas A&M University, College Station, TX, August 2003.

[22] A. Ambike, *Closed-Loop Real-Time Control on Distributed Networks*, M.S. Thesis, Texas A&M University, College Station, TX, August 2004.

[23] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, 3rd Ed., Reading, MA: Addison-Wesley, 1998.

[24] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 3rd Ed., Upper Saddle River, NJ: Prentice Hall 1997.

[25] Y. Halevi and A. Ray, "Integrated communication and control systems: Part I—analysis," *Journal of Dynamic Systems, Measurement and Control*, vol. 110, no. 4, pp. 367–373, December 1988.

[26] J. Nilsson, *Real-Time Control Systems with Delays*, Ph.D dissertation, Dept. Automatic Control. Lund Institute of Technology, Lund, Sweden, January 1998.

[27] L.A. Montestruque and P. J. Antsaklis, *Model-Based Networked Control Systems Stability*, ISIS Technical Report ISIS-2002-001, 2001.

[28] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 84–99, February 2001.

[29] G. C. Walsh and Y. Hong, and L.G. Bushnell, "Stability analysis of networked control system," *IEEE Transactions on Control System Technology*, vol. 10, no. 3, pp. 438–446, May 2002.

[30] F.-L. Lian and J. R. Moyne, "Modelling and optimal controller design of networked control systems with multiple delays," *International Journal of Control*, vol. 76, no. 6, pp. 591–606, April 2003.

[31] Z. Artstein, "Linear systems with delayed control: A reduction," *IEEE Transactions on Automatic Control,* vol. 27, no. 4, pp. 869–879, August 1982.

[32] L.-W. Liou and A. Ray, "A stochastic regulator for integrated communication and control systems: Part I – formulation of control law," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 113, no. 4, pp. 604–611, December, 1991.

[33] A. Ray, "Output feedback control under randomly varying distributed delays," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 4, pp. 701–711, August 1994.

[34] R. Luck and A. Ray, "An observer-based compensator for distributed delays," *Automatica*, vol. 26, no. 5, pp. 903–908, December 1990.

[35] H. Chan and U. Ozguner, "Closed-loop control of systems over a communications network with queues," *International Journal of Control*, vol. 62, no.3, pp. 493–510, June 1995.

[36] B. Marinescu and H. Bourlès, "Robust state-predictive control with separation property: A reduced-state design for control systems with non-equal time delays," *Automatica*, vol. 36, no. 4, pp. 555–562, April 2000.

[37] P. Seiler and R. Sengupta "Analysis of communication losses in vehicle control

problems," in *Proc. of 2001 American Control Conference*, Arlington, VA, pp. 1491–1496, June 2001.

[38] F.-L. Lian, J. Moyne, and D. Tilbury, "Network design consideration for distributed control systems," *IEEE Transactions on Control System Technology*, vol. 10, no. 2. pp. 297–307, March 2002.

[39] C. Foias, A. Tannenbaum, and G. Zames, "Weighted sensitivity minimization for delayed systems," *IEEE Transactions on Automatic Control*, vol. 31, no. 8, pp. 763–766, August 1986.

[40] B. Van Keulen, *H^{∞}-Control for Distributed Parameter Systems: A State-Space Approach*, Birkhauser, Basal: 1993.

[41] A. Kojima, K. Uchida, E. Shimemura, and S. Ishijima, "Robust stabilization of a system with delays in control," *IEEE Transactions on Automatic Control*, vol. 39, no. 8, pp. 1694–1698, August 1994.

[42] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Philadelphia, PA: SIAM, 1994.

[43] S. I. Niculescu, "H_{∞} memoryless control with an α stability constraint for time-delays systems: an LMI approach," in *Proc. of the 34th Conference on Decision and Control*, New Orleans, LA, pp. 1507–1512, December 1995.

[44] P. G. Otanez, J. R. Moyne, and D. M. Tilbury, "Using deadbands to reduce communication in networked control systems," in *Proc. of 2002 American Control Conference*, Anchorage, Alaska, pp. 3015–3020, May 2002.

[45] J. K. Yook, D. M. Tilbury, N. R. Soparkar, "Trading computation for bandwidth: reducing communication in distributed control system using state estimators," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 4, pp. 503–518, July 2002.

[46] S.-K. Kweon, K. G. Shin, and G. Workman, “Achieving real-time communication over Ethernet with adaptive traffic smoothing,” in *Proc. of IEEE Real-Time Technology Application Symposium*, Vancouver, Canada, pp. 90–100, June 1999.

[47] S. H. Hong, “Scheduling algorithm of data sampling times in the integrated communication and control systems,” *IEEE Transactions on Control Systems Technology*, vol. 3, no. 2, pp. 225–230, June 1995.

[48] S. H. Hong and Y. C. Kim, “Implementation of a bandwidth allocation scheme in a token-passing Fieldbus network,” *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 2, pp. 246–251, April 2002.

[49] H. S. Park, Y. H. Kim, D.-S. Kim, and W. H. Kwon, “A scheduling method for network-based control systems,” *IEEE Transactions on Control Systems Technology*, vol.10, no. 3, pp. 318–330, May 2002.

[50] M. S. Branicky, S. M. Phillips, and W. Zhang “Scheduling and feedback co-design for networked control systems,” in *Proc. of the IEEE Conference on Decision and Control*, Las Vegas, NV, pp. 1211–1217, December 2002.

[51] A. Cervin, J. Eker, B. Bernhardsson and K.-E. Årzén, “Feedback-feedforward scheduling of control tasks,” *Journal of Real-Time Systems*, vol. 23, no. 1, pp. 25–53, July 2002.

[52] J. Yepez, P. Marti and J. M. Fuertes, “Control loop scheduling paradigm in distributed control systems,” in *Proc. of the 29th IECON*, Roanoke, VA, pp. 1441–1446, November 2003.

[53] G. C. Walsh, and H. Ye, “Scheduling of networked control systems,” *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 57–65, February 2001.

[54] H. Rehbinder, M. Sanfridson. “Scheduling of a limited communication channel for

optimal control," *Automatica*, vol. 40, no. 3, pp. 491–500, March 2004.

[55] M. Velasco, J. M. Fuertes, C. Lin, P. Martí, and S. Brandt, "A control approach to bandwidth management in networked control systems," in *Proc. of 30th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2343–2348, November 2004.

[56] M. Velasco, P. Martí and M. Frigola, "Bandwidth management for distributed control of highly articulated robots," in *Proc. of IEEE International Conference on Robotics and Automation*, Barcelona, Spain, pp. 266–271, April 2005.

[57] F.-L. Lian, J. R. Moyne, and D. M. Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet and DeviceNet," *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 66–83, February 2001.

[58] A. Kutlu, H, Ekiz, and E. T. Powner, "Performance analysis of MAC protocols for wireless control area network," in *Proc. of the Second International Symposium on Parallel Architectures, Algorithms, and Networks*, Beijing, China, pp. 494–499, June 1996.

[59] B. Englert, L. Rudolph, and A. Shvartsman, "Developing and refining an adaptive token-passing strategy," in *Proc. of the 21st International Conference on Distributed Computing Systems*, Phoenix, AZ, pp. 597–605, April 2001.

[60] A. S. Tanenbaum, *Computer Networks*, 3rd Ed., Upper Saddle River, NJ: Prentice-Hall, 2001.

[61] S.-K. Kweon, K. G. Shin, and G. Workman, "Achieving real-time communication over Ethernet with adaptive traffic smoothing," in *Proc. of Real-Time Technology and Applications Symposium*, Vancouver, Canada, pp. 90–100, June 1999.

[62] C. Venkatramani and T. Chiueh, "Supporting real-time traffic on Ethernet," in *Proc. of Real-Time Systems Symposium*, San Juan, Puerto Rico, pp. 282–286, December 1994.

[63] H. Ye, G. Walsh, and L. Bushnell, "Wireless local area networks in the manufacturing industry," in *Proc. of 2000 American Control Conference*, Chicago, IL, pp. 2363–2367, June 2000.

[64] H. Ye and G. Walsh, "Real-time mixed-traffic wireless networks," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 5, pp. 883–890, October 2001.

[65] Decotignie, J.D., "A perspective on Ethernet - TCP/IP as a Fieldbus," in *Proc. of IFAC Conference on Fieldbus Systems*, Nancy, France, pp. 23–28, 2001.

[66] T. Skeie, S. Johannessen, and C. Brunner, "Ethernet in substation automation," *IEEE Control System Magazine*, vol. 22, no. 3, pp. 43–51, June 2002.

[67] S. C. Paschall, II, *Design, Fabrication and Control of a Single Actuator Magnetic Levitation System*, Senior Honors Thesis, Texas A&M University, College Station, TX, May 2002.

[68] B. Lee and J.-G. Lee, "Robust stability and stabilization of linear delayed systems with structured uncertainty," *Automatica*, vol. 35, no. 6, pp. 1149–1154, June 1999.

[69] X. Li, M. Guay, B. Huang, and D. G. Fisher, "Delay-dependent robust H_∞ control of uncertain linear systems with input delay," in *Proc. of 1999 American Control Conference*, San Diego, CA, pp. 800–804, June 1999.

[70] D. A. Mellichamp, *Real-Time Computing with Applications to Data Acquisition and Control*, 1st ed., Ontario, Canada: Van Nostrand Reinhold Company, 1983.

[71] T. Boutell, *CGI Programming in C and Perl*, New York: Addison-Wesley Developers Press, 1995.

[72] A. Ambike., W. -J. Kim, and K. Ji, "Real-time operating environment for networked control systems," in *Proc. of 2005 American Control Conference*, Portland, Oregon, pp. 2353–2358, June 2005.

[73] P. Mantegazza, “DIAPM RTAI - real-time application,” [Online]. Available: http://www.rtai.org, May 2004.

[74] R. A. Volz, *Real-Time Computing*, Lecture notes for CPSC 456, Department of Computer Science, Texas A&M University, College Station, TX, 2003.

[75] D. Schleef, “Linux control and measurement device interface,” [Online]. Available: http://www.comedi.org, May 2004.

[76] S. Tatham, O. Dunn, B. Harris, and J. Nevins, “PuTTY: A free Telnet/SSH client,” [Online]. Available: http://www.chiark.greenend.org.uk/~sgtatham/putty, July 2005.

[77] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. New York: Wiley, 1996.

[78] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 3rd Ed., New York: Addison-Wesley, 1994.

[79] L. Zhang, Z. Liu, and C. H. Xia, “Clock synchronization algorithms for network measurements,” in *Proc. of IEEE INFOCOM*, New York, NY, vol. 1, pp. 160–169, June 2002.

[80] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, The MIT Press, Cambridge, MA, 1983.

[81] Y. Wang, L. Xie, and C. E. de Souza, “Robust control of a class of uncertain nonlinear systems,” *Systems & Control Letters*, vol. 19, no. 2, pp.139–149, August 1992.

[82] C. M. Krishina and K. G. Shin, *Real-Time Systems*, New York: McGraw-Hill, 1997.

[83] K. J. Astrom, *Introduction to Stochastic Control Theory*, Acadimic Press, New York: 1970.

[84] S. S. Hu and Q. X. Zhu, “Stochastic optimal control and analysis of stability of networked control systems with long delay,” *Automatica*, vol. 39, no. 11, pp. 1877–1884,

November 2003.

[85] H.-J. Yoo and O.-K. Kwon, “Design of networked control system for multiple plants with time varying delays,” in *Proc. of the 23rd IASTED International Conference of Modeling, Identification, and Control*, pp. 44–49, Grindelwald, Switzerland, February 2004.

[86] P. Marti, J. Yepez, M.Velasco, R. Villa, and J.M. Fuertes, “Managing quality-of-control in network-based control systems by controller and message scheduling co-design,” *IEEE Transactions on Industrial Electronics*, vol. 51, no. 6, pp. 1159–1167, December 2004.

[87] A. Bar-Noy, B. Patt-Shamir, and I. Ziper, “Broadcast disks with polynomial cost functions,” in *Proc. of the 19th Joint Conference of the IEEE Computer and Communication Societies*, pp. 575–584, Tel-Aviv, Israel, March 2000.

[88] M.-L. Shyu, S.-C. Chen, and H. Luo, “Optimal bandwidth allocation scheme with delay awareness in multimedia transmission,” in *Proc. of IEEE International Conference on Multimedia and Exposition*, Lausanne, Switzerland, pp. 537–540, August 2002.

APPENDIX A

DISCRETE-TIME ARTSTEIN TRANSFORM

The continuous-time system equation is given by

$$\dot{x}(t) = A_c x(t) + B_c u(t-\tau). \tag{A.1}$$

Then the Artstein transform [31] in this case is

$$\widetilde{x}(t) = x(t) + \int_{-\tau}^{0} e^{-A_c(\tau+s)} B_c u(t+s) ds\,. \tag{A.2}$$

The discrete-time system is

$$x(k+1) = Ax(k) + Bu(k-N)\,, \tag{A.3}$$

where h is the sampling period, $N = \tau / h$, and

$$A = e^{A_c h}, B = \left(\int_0^h e^{A_c s} ds \right) B_c\,. \tag{A.4}$$

Writing (A.3) at $t = kh$, $s = ih$ and from (A.4) we obtain

$$\widetilde{x}(kh) = x(kh) + \int_{-Nh}^{0} e^{-A_c h(i+N)} B_c u(kh+ih) d(ih)$$

$$= x(kh) + \sum_{i=-N}^{-1} e^{=A_c h(i+N+1)} e^{A_c h} B_c u(kh+ih) h$$

$$= x(kh) + \sum_{i=-N}^{-1} e^{-A_c h(i+N+1)} (e^{A_c h} h B_c) u(kh+ih)$$

$$= x(kh) + \sum_{i=-N}^{-1} A^{-(i+N+1)} B u(kh+ih) \quad .$$

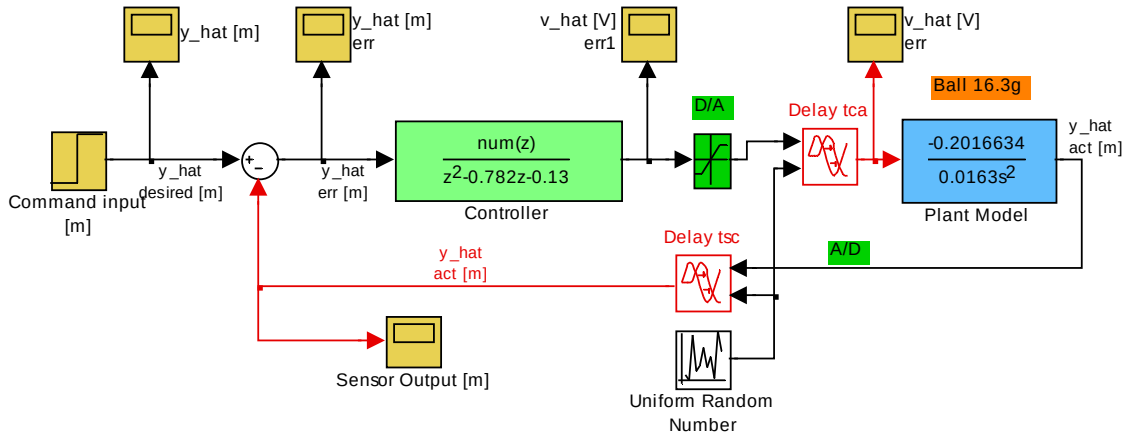
Thus

$$\widetilde{x}(k) = x(k) + \sum_{i=-N}^{-1} A^{-(i+N+1)} B u(k+i) \, . \tag{A.5}$$

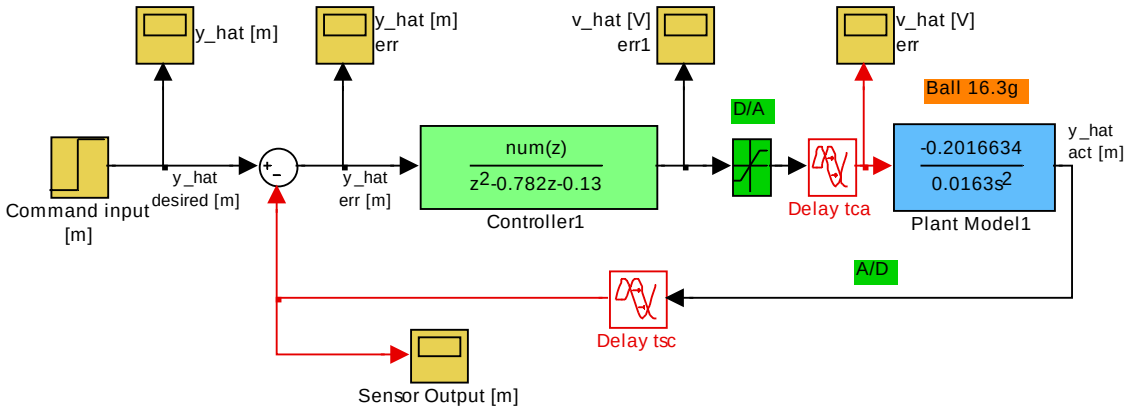
APPENDIX B

MATLAB/SIMULINK® CODES

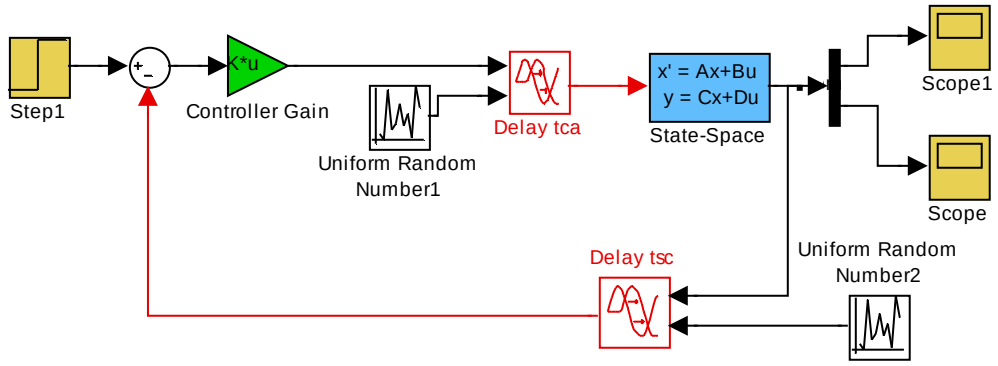
B.1 Block Diagram of the Ball Maglev System Simulation with Random Delays



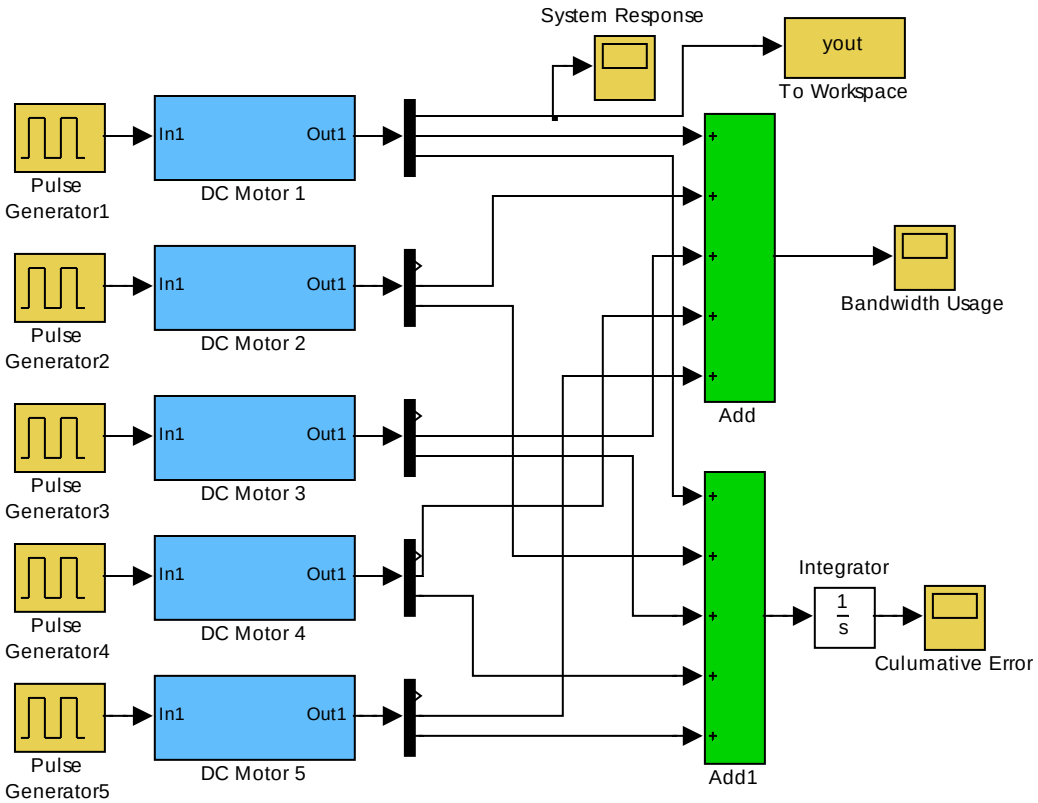
B.2 Block Diagram of the Ball Maglev System Simulation with Constant Delays



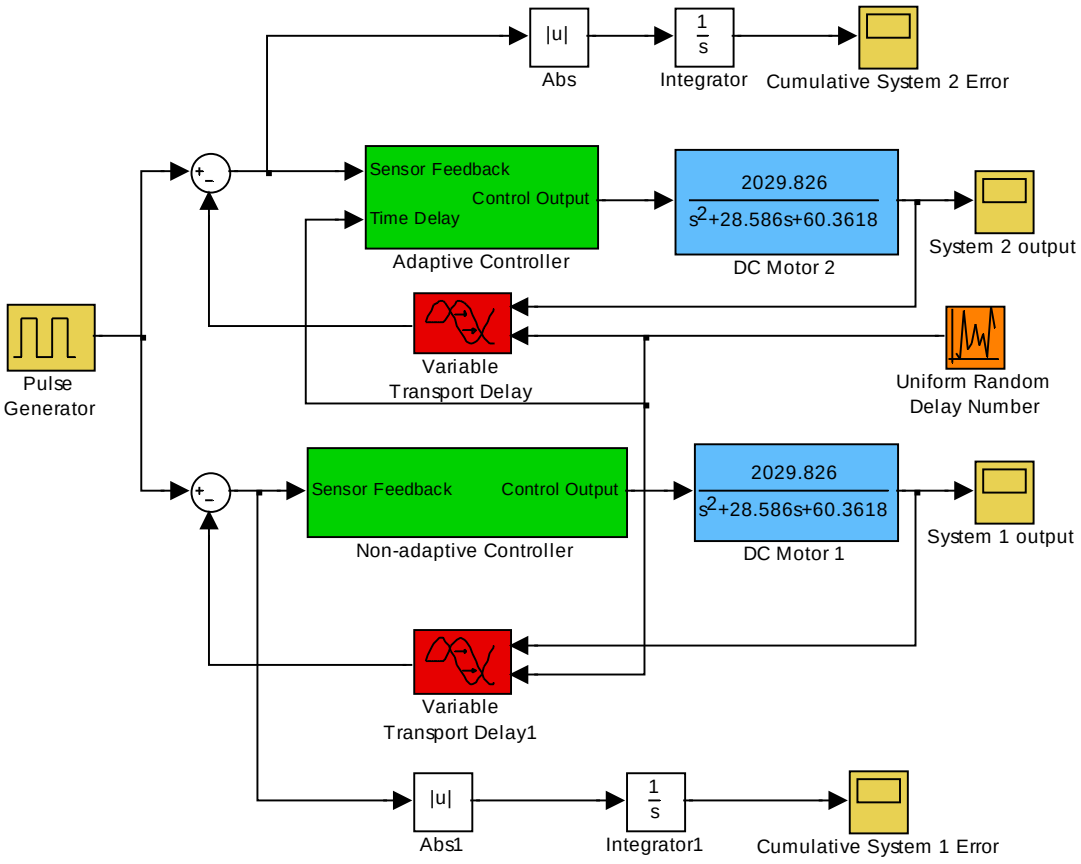
B.3 Block Diagram of the Simulation of the Numerical System (5.52)



B.4 Block Diagram of the Simulation of the NCSs with 5 DC Motors in Chapter VI



B.5 Block Diagram of the Simulation of the Adaptive Controller in Chapter VI



APPENDIX C

C CODES FOR CLOSED-LOOP CONTROL OVER THE ETHERNET

C. 1 C Codes for Closed-loop Control without Packet Loss

Client Side:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <pthread.h>
#include <signal.h>
#include <comedilib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/ioctl.h>
#include "defines.h"
#define KEEP_STATIC_INLINE
#include <rtai_lxrt_user.h>
#include <rtai_lxrt.h>
#define PERIOD 1000000
#define LOOPS 1000
#define NTASKS 2
#define taskname(x) (1000 + (x))
RTIME time_stamp;
```

```
double u0;
double y0;
double y_1;
double y_2;
double u_1;
double u_2;
RTIME current_time_stamp;
pthread_t task[NTASKS];
int ntasks = NTASKS;
RT_TASK *mytask;
SEM *sem;
static int cpus_allowed;
SEM *sock_sem;    //* socket semaphoere, used by all the threads.
int sockid;
RTIME start_instant;
int server_sock_size = 0;
struct sockaddr_in my_addr, server_addr;
comedi_t *it;
int in_subdev = 0;
int out_subdev = 1;
int in_chan = 0;
int out_chan = 0;
int in_range = 0;
int out_range = 0;
int aref = AREF_GROUND;
int i=0;
//*comedi declarations
lsampl_t in_data;
lsampl_t out_data;
float volts = 0.0;
int in_maxdata = 0, out_maxdata = 0;
comedi_range *in_range_ptr, *out_range_ptr;
```

```
int endme_int = 0;

void terminate_normally(int signo);

void endme(int sig)
{
printf("You want to kill me?\n");
endme_int = 1;
exit(1);
}

void *send_thread_fun(void *arg)
{
RTIME start_time, period, end_time, difference;
RTIME t0;
SEM *sem;
RT_TASK *mytask;
unsigned long mytask_name;
int mytask_indx;
double * buffer = NULL;
int iRet = 0;
struct recv_data *send_msg = NULL;
int send_msg_size;
pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
mytask_indx = 0;
mytask_name = taskname(mytask_indx);
cpus_allowed = 1 - cpus_allowed;
if (!(mytask = rt_task_init_schmod(mytask_name, 1, 0, 0, SCHED_FIFO, 1 << cpus_allowed)))
printf("CANNOT INIT send_thread TASK\n");
exit(1);
}
printf("send thread pid = %ld\t master pid = %ld\n", getpid(), getppid());
```

```
mlockall(MCL_CURRENT | MCL_FUTURE);
rt_receive(0, (unsigned int*)&sem);
send_msg_size = sizeof(struct recv_data);
if(( send_msg = (struct recv_data *)calloc(1, sizeof(struct recv_data))) == NULL)
{
printf("cannot allocate message memory\n");
exit(4);
}
period = nano2count(PERIOD);
start_time = rt_get_time() + nano2count(10000000);
t0 = start_instant;
printf("send: t0 = %lld\t", t0);
printf("This period = %lld\t", rt_get_time());
printf("actual start = %lld\n", t0 + nano2count(500000000));
rt_task_make_periodic(mytask, (t0 + nano2count(500000000)), nano2count(3000000));
start_time = rt_get_cpu_time_ns();
printf("starting the send_thread while loop\n");
for(;;)
{
if(endme_int == 1)
{
break;
}
comedi_data_read(it, in_subdev, in_chan, in_range, aref, &in_data);
if(in_data > 4094)
{
in_data = 4094;
}
if(in_data < 2049)
{
in_data = 2049;
}
```

```
current_time_stamp = rt_get_cpu_time_ns();
y0 = comedi_to_phys(in_data, in_range_ptr, in_maxdata);
send_msg->y0 = y0;
send_msg->y_1 = y_1;
send_msg->y_2 = y_2;
send_msg->u_1 = u_1;
send_msg->u_2 = u_2;
send_msg->time_stamp = current_time_stamp;

rt_sem_wait(sock_sem);
iRet = sendto(sockid, (const void *)send_msg, send_msg_size, 0, (structsockaddr*)&server_addr,
server_sock_size);
rt_sem_signal(sock_sem);
if(iRet <= -1)
{
perror("sendto() failed\n");
break;
}
y_2 = y_1;
y_1 = y0;
printf("volts=%f\t time_stamp=%lld\n", volts, current_time_stamp);
rt_task_wait_period();
}
end_time = rt_get_cpu_time_ns();
endme_int++;
rt_sem_signal(sem);
rt_make_soft_real_time();
free(send_msg);
rt_task_delete(mytask);
printf("send_thread ENDS\n");
return 0;
}
```

```
void *recv_thread_fun(void *arg)
{
RTIME start_time, period, end_time, difference;
RTIME t0;
SEM *sem;
RT_TASK *mytask;
unsigned long mytask_name;
int mytask_indx;
struct data *buffer = NULL;
int iRet = 0;
int recv_msg_size;
struct send_data *recv_msg = NULL;
recv_msg_size = sizeof(struct send_data);
if(( recv_msg = (struct send_data *)calloc(1, sizeof(struct send_data))) == NULL)
{
printf("cannot allocate message memory\n");
exit(4);
}
pthread_setcanceltype(PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
mytask_indx = 1;
mytask_name = taskname(mytask_indx);
cpus_allowed = 1 - cpus_allowed;
if (!(mytask = rt_task_init_schmod(mytask_name, 1, 0, 0, SCHED_FIFO, 1 << cpus_allowed)))
{
printf("CANNOT INIT recv_thread TASK\n");
exit(1);
}
mytask_name, mytask);
printf("recv thread pid = %ld\t master pid = %ld\n", getpid(), getppid());
mlockall(MCL_CURRENT | MCL_FUTURE);
rt_receive(0, (unsigned int*)&sem);
period = nano2count(PERIOD);
```

```
start_time = rt_get_time() + nano2count(10000000);
t0 = start_instant;
printf("recv: t0 = %lld\t", count2nano(t0));
printf("This period = %lld\t", count2nano(rt_get_time()));
printf("actual start = %lld\n", count2nano(t0 + nano2count(500500000)));
rt_task_make_periodic(mytask, (t0 + nano2count(500500000)), nano2count(3000000));
start_time = rt_get_time();
printf("starting the recv_thread while loop\n");
for(;;)
{
if(endme_int == 1)
{
break;
}
rt_sem_wait(sock_sem);
while(1)
{
iRet = recvfrom(sockid, (void *)recv_msg, recv_msg_size, 0,
(struct sockaddr *)&server_addr, &server_sock_size);
if(iRet < 1)
{
break;
}
}
rt_sem_signal(sock_sem);
if(iRet <= -1)
{
endme_int = 1;
perror("recvfrom() failed\n");
break;
}
u0 = recv_msg->u0;
```

```
volts = u0;
if(volts > 10.0)
{
volts = 9.99999;
}
if(volts < 0.0)
{
volts = 0.0;
}
out_data = comedi_from_phys(volts, out_range_ptr, out_maxdata);
comedi_data_write(it, out_subdev, out_chan, out_range, aref, out_data);
u_2 = u_1;
u_1 = u0;
rt_task_wait_period();
}
end_time = rt_get_cpu_time_ns();
endme_int++;
rt_make_soft_real_time();
free(recv_msg);
rt_task_delete(mytask);
printf("recv_thread ENDS\n");
return 0;
}

int main(void)
{
int i;
unsigned long mytask_name = nam2num("MASTER");
struct sigaction sa;
char * server_ip = "165.91.214.188";
unsigned short my_port, server_port;
my_port = 4445;
```

```
server_port = 4444;
/* -- Create client side socket -- */
printf("creating socket\n");
if( (sockid = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
perror("socket() failed ");
exit(2);
}
/* -- Initialize client side socket address -- */
memset((void *) &my_addr, (char) 0, sizeof(my_addr));
my_addr.sin_family = AF_INET;                    /* Internet Address Family */
my_addr.sin_addr.s_addr = htonl(INADDR_ANY);  /* I can receive from any host */
my_addr.sin_port = htons(my_port);
if ( (bind(sockid, (struct sockaddr *) &my_addr, sizeof(my_addr)) < 0) )
{
perror("bind() failed ");
exit(3);
}
/* -- Initialize server side socket address -- */
server_sock_size = sizeof(server_addr);
memset((void *) &server_addr, (char) 0, server_sock_size);
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = inet_addr(server_ip);
server_addr.sin_port = htons(server_port);
sa.sa_handler = endme;
sa.sa_flags = 0;
sigemptyset(&sa.sa_mask);
if(sigaction(SIGINT, &sa, NULL))
{
perror("sigaction");
}
if(sigaction(SIGTERM, &sa, NULL))
```

```
{
perror("sigaction");
}
it = comedi_open("/dev/comedi0");
if(it == NULL)
{
printf("Could not open comedi\n");
exit(1);
}
in_maxdata = comedi_get_maxdata(it, in_subdev, in_chan);
out_maxdata = comedi_get_maxdata(it, out_subdev, out_chan);
in_range_ptr = comedi_get_range(it, in_subdev, in_chan, in_range);
out_range_ptr = comedi_get_range(it, out_subdev, out_chan, out_range);
if (!(mytask = rt_task_init(mytask_name, 1, 0, 0))) {
printf("CANNOT INIT main TASK \n");
exit(1);
}
printf("MASTER INIT: name = %lu, address = %p.\n", mytask_name, mytask);
sem = rt_sem_init(10000, 0);
sock_sem = rt_sem_init(nam2num("SOCK"), 1);
rt_set_periodic_mode();
start_rt_timer(nano2count(25000));
start_instant = rt_get_time();
printf("main: start_instant = %lld\n", start_instant);
if (pthread_create(&task[0], NULL, send_thread_fun, &start_instant))
{
printf("ERROR IN CREATING send_thread\n");
exit(1);
}
if (pthread_create(&task[1], NULL, recv_thread_fun, &start_instant))
{
printf("ERROR IN CREATING recv_thread\n");
```

```
exit(1);
}
for (i = 0; i < ntasks; i++)
{
while (!rt_get_adr(taskname(i)))
{
rt_sleep(nano2count(20000000));
}
}
for (i = 0; i < ntasks; i++)
{
rt_send(rt_get_adr(taskname(i)), (unsigned int)sem);
}
printf("Start waiting for sem\n");
while(endme_int == 0)
{
rt_sem_wait_timed(sem, nano2count(5000000000));
}
printf("Stop waiting for sem\n");
for (i = 0; i < ntasks; i++)
{
while (rt_get_adr(taskname(i)))
{
rt_sleep(nano2count(20000000));
}
}
rt_sem_delete(sem);
rt_sem_delete(sock_sem);
stop_rt_timer();
comedi_close(it);
rt_task_delete(mytask);
printf("MASTER %lu %p ENDS\n", mytask_name, mytask);
```

```
for (i = 0; i < ntasks; i++) {
pthread_join(task[i], NULL);
}
return 0;
}
```

Server Side:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <asm/errno.h>
#include <sys/types.h>
#include <sys/user.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sched.h>
#include <comedilib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <errno.h>
#include <inttypes.h>
#include "defines.h"
#define KEEP_STATIC_INLINE
#include <rtai_lxrt_user.h>
```

```
#include <rtai_lxrt.h>
RTIME time_stamp;
double u0;
double y0;
double y_1;
double y_2;
double u_1;
double u_2;
double y_hat_desi = 0.0;          //* User input (desired set point)
double v_hat_err = 0.0;
double k = 0.098;                 //* Gain parameter
double c = 0.0;                   //* Controller constant
double v = 1.018;                 //* For initial offset
double er0 = 0.0;                 //* Input to controller at time n
double er1 = 0.0;                 //* Intput to controller at time n-1
double er2 = 0.0;                 //* Intput to controller at time n-2
int i=0;
                                  //*rtai declarations
unsigned long mtsk_name;
RT_TASK *mtsk;
struct sched_param mysched;
size_t readn(int fd, void *vptr, size_t n)
{
size_t nleft;
size_t nread;
char *ptr;
ptr = vptr;
nleft = n;
while(nleft > 0)
{
if((nread = read(fd, ptr, nleft)) < 0)
{
```

```
return -1;
}
else if(nread == 0)
{
break;
}
nleft -= nread;
ptr += nread;
}
return (n - nleft);
}
size_t writen(int  fd, const void *vptr, size_t n)
{
size_t nleft;
size_t nwritten;
const char *ptr;
ptr = vptr;
nleft = n;
while(nleft > 0)
{
if((nwritten  = write(fd, ptr, nleft)) <= 0)
{
return -1;
}
nleft -= nwritten;
ptr += nwritten;
}
return n;
}

void terminate_normally(int signo)
{
```

```
fflush(stdin);
if(signo==SIGINT || signo==SIGTERM)
{
printf("Terminating the program normally\n");
rt_make_soft_real_time();
printf("MASTER TASK YIELDS ITSELF\n");
rt_task_yield();
printf("MASTER TASK STOPS THE PERIODIC TIMER\n");
stop_rt_timer();
printf("MASTER TASK DELETES ITSELF\n");
rt_task_delete(mtsk);
printf("END MASTER TASK\n");
}
exit(0);
}

main(int argc, char *argv[])
{
int sockid, nread, addrlen;
struct sockaddr_in my_addr, client_addr;
int nw, nr;
int send_buffer_size, recv_buffer_size;
unsigned short server_port = 0;
struct send_data *send_buffer = NULL;
struct recv_data *recv_buffer = NULL;
RTIME start_time = 0;
RTIME end_time = 0;
RTIME actual_period = 0;
RTIME difference = 0;
size_t iRet = 0;
int esti_count = 0;
double vhaterr_prev[5] = {0.0, 0.0, 0.0, 0.0, 0.0};
```

```
int j=0;
//*signal handling
struct sigaction sa;
//*Initialize the signal handling structure
sa.sa_handler = terminate_normally;
sa.sa_flags = 0;
sigemptyset(&sa.sa_mask);
if(sigaction(SIGINT, &sa, NULL))
{
perror("sigaction");
}
if(sigaction(SIGTERM, &sa, NULL))
{
perror("sigaction");
}
fprintf(stderr, "creating socket\n");
if ( (sockid = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
perror("socket() failed ");
fprintf(stderr, "%s: socket error: %d\n", argv[0], errno);
exit(2);
}
fprintf(stderr, "binding my local socket\n");
server_port = 4444;
memset((void *) &my_addr, (char) 0, sizeof(my_addr));
my_addr.sin_family = AF_INET;
my_addr.sin_addr.s_addr = htons(INADDR_ANY);
my_addr.sin_port = htons(server_port);
if ( (bind(sockid, (struct sockaddr *) &my_addr,
sizeof(my_addr)) < 0) )
{
perror("bind() failed ");
```

```
fprintf(stderr, "bind() errno = %d\n", errno);
exit(4);
}
recv_buffer_size = sizeof(struct recv_data);
if(( recv_buffer = (struct recv_data *)calloc(1, sizeof(struct recv_data))) ==NULL)
{
fprintf(stderr, "cannot allocate memory for buffer!\n");
exit(4);
 }
send_buffer_size = sizeof(struct send_data);
if(( send_buffer = (struct send_data *)calloc(1, sizeof(struct send_data))) ==NULL)
{
fprintf(stderr, "cannot allocate memory for buffer!\n");
exit(4);
}
addrlen = sizeof(client_addr);
fprintf(stderr, "%s: starting blocking message read\n", argv[0]);
mysched.sched_priority = 99;
if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 )
{
puts(" ERROR IN SETTING THE SCHEDULER UP");
perror( "errno" );
exit( 0 );
}
mlockall(MCL_CURRENT | MCL_FUTURE);
mtsk_name = nam2num("MTSK");
if (!(mtsk = rt_task_init(mtsk_name, 0, 0, 0))) {
printf("CANNOT INIT MASTER TASK\n");
exit(1);
}
start_time = rt_get_cpu_time_ns();
printf("main: start_time = %lld\n", start_time);
```

```
printf("MASTER TASK STARTS THE ONESHOT TIMER\n");
actual_period = start_rt_timer(nano2count(25000));
printf("actual_period = %lld\n", actual_period);
printf("MASTER TASK MAKES ITSELF PERIODIC \n");
rt_task_make_periodic(mtsk, rt_get_time()+ nano2count(3000000), nano2count(3000000));
while( 1 )
{
start_time = rt_get_cpu_time_ns();
nr = recvfrom(sockid, (void *)recv_buffer, recv_buffer_size, 0, (struct sockaddr *) &client_addr,
&addrlen);
if( nr <= -1 )
{
fprintf(stderr, "recvfrom() errno = %d\n", errno);
exit(10);
}
y0 = recv_buffer->y0;
y_1=recv_buffer->y_1;
y_2=recv_buffer->y_2;
u_1 = recv_buffer->u_1;
u_2 = recv_buffer->u_2;
er0 = (y_hat_desi - (-0.0004653*y0+0.002525))*k;   //* Error Calculation *//
er1 = (y_hat_desi - (-0.0004653*y_1+0.002525))*k;
er2 = (y_hat_desi - (-0.0004653*y_2+0.002525))*k;
u0 = ((0.782*(u_1-v)) + (0.13*(u_2-v)) - (41500.0*er0) + (41500.0*1.754*er1) -
(41500.0*0.769*er2)) + v;
send_buffer->u0 = u0;
send_buffer->time_stamp = recv_buffer->time_stamp;
nw = sendto(sockid, (const void *)send_buffer, send_buffer_size, 0, (struct sockaddr *)
&client_addr, addrlen);
 if( nw <= -1 )
{
perror("sendto failed ");
```

```
fprintf(stderr, "sendto() errno = %d \n", errno);
exit(12);
}
start_time = rt_get_cpu_time_ns();
 }
printf("MASTER TASK YIELDS ITSELF\n");
rt_task_yield();
printf("MASTER TASK STOPS THE PERIODIC TIMER\n");
stop_rt_timer();
printf("MASTER TASK DELETES ITSELF\n");
rt_task_delete(mtsk);
close(sockid);
free(send_buffer);
free(recv_buffer);
}
```

C. 2 C Codes for Closed-loop Control with Packet Losses

C.2.1 Stabilization with 4 Consecutive Packet Losses

(1) Model-estimation-based Compensation Algorithm

Client Side:

```
 #include <stdio.h>
…                                  //* "…" means identical codes with the codes in Appendix C.1
double u1e;
double u2e;
double u3e;
double u4e;
double U1;
double U2;
double U3;
```

```
double U4;
…
long indi = 0;
…

void *send_thread_fun(void *arg)
{
...
for(;;)
{
…
send_msg->y0 = y0;
…
}
…
}

void *recv_thread_fun(void *arg)
{
…
for(;;)
{
indi ++;
…
u0 = recv_msg->u0;
u1e = recv_msg->u1e;
u2e = recv_msg->u2e;
u3e = recv_msg->u3e;
u4e = recv_msg->u4e;
…
volts = u0;
if (indi = 1800)                    //* 4 packet losses occurring once every 6 s
```

```
{
U1 = u1e;
U2 = u2e;
U3 = u3e;
U4 = u4e;
}
if (indi = 1801)
{
volts =U1;
}
if (indi = 1802)
{
volts =U2;
}
if (indi = 1803)
{
volts =U3;
}
if (indi = 1804)
{
volts =U4;
indi = 0;
}
}
…
}

int main(void)
{
…
printf("Ball Position = %f mm\n", volts);
}
```

Server Side:

```
#include <stdio.h>
…
double u1e;        //*estimated control data
double u2e;
double u3e;
double u4e;
…
double y_hat_1;
double y_hat_2;
double y_hat_3;
double y_hat_4;
…

main(int argc, char *argv[])
{
…
while( 1 )
{
…
y_hat_1 = y0;                                   //* model estimation
u1e = 0.78*u0 + 0.13*u_1 -18.92*y_hat_1+33.19*y0 -15.06*y_1+0.24;
send_buffer->u1e = u1e;
y_hat_2 = y_hat_1;
u2e = 0.78*u1e + 0.13*u0 -18.92*y_hat_2+33.19*y_hat_1 -15.06*y0+0.24;
send_buffer->u2e = u2e;
y_hat_3 = y_hat_2;
u3e = 0.78*u2e + 0.13*u1e -18.92*y_hat_3+33.19*y_hat_2 -15.06*y_hat_1+0.24;
send_buffer->u3e = u3e;
y_hat_4 =y_hat_3;
u4e = 0.78*u3e + 0.13*u2e -18.92*y_hat_4+33.19*y_hat_3 -15.06*y_hat_2+0.24;
send_buffer->u4e = u4e;
```

```
…
}
…
}
```

(2) Predictor-based Compensation Algorithm

Client Side:

```
#include <stdio.h>
…
double u1p;              //*predicted control data
double u2p;
double u3p;
double u4p;
double U1;
double U2;
double U3;
double U4;
…
double y_1;
double y_2;
double y_3;
double y_4;
double y_5;
double y_6;
double y_7;
long indi = 0;
…
void *send_thread_fun(void *arg)
{
...
```

```
for(;;)
{
…
send_msg->y0 = y0;
send_msg->y_1 = y_1;
send_msg->y_2 = y_2;
send_msg->y_3 = y_3;
send_msg->y_4 = y_4;
send_msg->y_5 = y_5;
send_msg->y_6 = y_6;
send_msg->y_7 = y_7;
…
y_7 = y_6;
y_6 = y_5;
y_5 = y_4;
y_4 = y_3;
y_3 = y_2;
y_2 = y_1;
y_1 = y0;
…
}
…
}

void *recv_thread_fun(void *arg)
{
…
for(;;)
{
indi ++;
…
u0 = recv_msg->u0;
```

```
u1p = recv_msg->u1p;
u2p = recv_msg->u2p;
u3p = recv_msg->u3p;
u4p = recv_msg->u4p;
…
volts = u0;
if (indi = 1800)                    //* 4 packet losses occurring once every 6 s
{
U1 = u1p;
U2 = u2p;
U3 = u3p;
U4 = u4p;
}
if (indi = 1801)
{
volts =U1;
}
if (indi = 1802)
{
volts =U2;
}
if (indi = 1803)
{
volts =U3;
}
if (indi = 1804)
{
volts =U4;
indi =0;
}
}
…
```

```
}

int main(void)
{
…
printf("Ball Position = %f mm\n", volts);
…
}
```

Serve Side:

```
#include <stdio.h>
…
double u1p;          //*predicted control data
double u2p;
double u3p;
double u4p;
…
double y_1;
double y_2;
double y_3;
double y_4;
double y_5;
double y_6;
double y_7;
…
double y_hat_1;
double y_hat_2;
double y_hat_3;
double y_hat_4;
…
```

```
main(int argc, char *argv[])
{
…
while( 1 )
{
…
y_1 = recv_buffer->y_1;
y_2 = recv_buffer->y_2;
y_3 = recv_buffer->y_3;
y_4 = recv_buffer->y_4;
y_5 = recv_buffer->y_5;
y_6 = recv_buffer->y_6;
y_7 = recv_buffer->y_7;
…                                                    //* sensor data prediction
y_hat_1 = 0.8122*y0 - 0.3479*y_1 - 0.0294*y_2 + 0.4605*y_3 + 0.0742*y_4 + 0.1042*y_5 +
0.1117*y_6 - 0.3561*y_7;
er0 = (y_hat_desi - (-0.0004653*y_hat_1+0.002525))*k;  //* Error Calculation
er1 = (y_hat_desi - (-0.0004653*y0+0.002525))*k;
er2 = (y_hat_desi - (-0.0004653*y_1+0.002525))*k;
u1p = ((0.782*(u0-v)) + (0.13*(u_1-v)) - (41500.0*er0) + (41500.0*1.754*er1) -
(41500.0*0.769*er2)) + v;
send_buffer->u1p = u1p;
y_hat_2 = 0.3117*y0 - 0.3119*y_1 + 0.4366*y_2 + 0.4482*y_3 + 0.1645*y_4 + 0.1964*y_5 -
0.2653*y_6 - 0.2892*y_7;
er0 = (y_hat_desi - (-0.0004653*y_hat_2+0.002525))*k;
er1 = (y_hat_desi - (-0.0004653*y_hat_1+0.002525))*k;
er2 = (y_hat_desi - (-0.0004653*y0+0.002525))*k;
u2p = ((0.782*(u1p-v)) + (0.13*(u0-v)) - (41500.0*er0) + (41500.0*1.754*er1) -
(41500.0*0.769*er2)) + v;
send_buffer->u2p = u2p;
y_hat_3 = -0.0587*y0 + 0.3281*y_1 + 0.4390*y_2 + 0.3080*y_3 + 0.2195*y_4 - 0.2329*y_5 -
0.2544*y_6 - 0.1110*y_7;
```

```
er0 = (y_hat_desi - (-0.0004653*y_hat_3+0.002525))*k;
er1 = (y_hat_desi - (-0.0004653*y_hat_2+0.002525))*k;
er2 = (y_hat_desi - (-0.0004653*y_hat_1+0.002525))*k;
u3p = ((0.782*(u2p-v)) + (0.13*(u1p-v)) - (41500.0*er0) + (41500.0*1.754*er1) -
(41500.0*0.769*er2)) + v;
send_buffer->u3p = u3p;
y_hat_4 = 0.2804*y0 + 0.4594*y_1 + 0.3097*y_2 + 0.1925*y_3 - 0.2372*y_4 - 0.2605*y_5 -
0.1176*y_6 + 0.0209*y_7;
er0 = (y_hat_desi - (-0.0004653*y_hat_4+0.002525))*k;
er1 = (y_hat_desi - (-0.0004653*y_hat_3+0.002525))*k;
er2 = (y_hat_desi - (-0.0004653*y_hat_2+0.002525))*k;
u4p = ((0.782*(u3p-v)) + (0.13*(u2p-v)) - (41500.0*er0) + (41500.0*1.754*er1) -
(41500.0*0.769*er2)) + v;
send_buffer->u4p = u4p;
…
}
…
}
```

C.2.2 Stabilization with 20% Packet Losses

(1) Model-estimation-based Compensation Algorithm

Client Side:

```
 #include <stdio.h>
…
double u1e;                    //* estimated control data
double u2e;
double u3e;
double u4e;
…
```

```
long indi = 0;
…

void *send_thread_fun(void *arg)
{
...
for(;;)
{
…
send_msg->y0 = y0;
send_msg->y_1 = y_1;
send_msg->y_2 = y_2;
…
}
…
}

void *recv_thread_fun(void *arg)
{
…
for(;;)
{
indi ++;
…
u0 = recv_msg->u0;
u1e = recv_msg->u1e;
u2e = recv_msg->u2e;
u3e = recv_msg->u3e;
u4e = recv_msg->u4e;
…
volts = u0;
if (indi = 5)                    //* 20% packet losses
```

```
{
volts =u1e;
indi = 0;
}
}
…
}

int main(void)
{
…
fprintf(fp, “%f\n”, volts);
printf(“Ball Position = %f mm\n”, volts);
…
}
```

(2) Predictor-based Compensation Algorithm

Client Side:

```
#include <stdio.h>
…
double u1p;
double u2p;
double u3p;
double u4p;
…
double y_1;
double y_2;
double y_3;
double y_4;
double y_5;
```

```
double y_6;
double y_7;
long indi = 0;
…

void *send_thread_fun(void *arg)
{
...
for(;;)
{
…
send_msg->y0 = y0;
send_msg->y_1 = y_1;
send_msg->y_2 = y_2;
send_msg->y_3 = y_3;
send_msg->y_4 = y_4;
send_msg->y_5 = y_5;
send_msg->y_6 = y_6;
send_msg->y_7 = y_7;
…
y_7 = y_6;
y_6 = y_5;
y_5 = y_4;
y_4 = y_3;
y_3 = y_2;
y_2 = y_1;
y_1 = y0;
…
}
…
}
```

```
void *recv_thread_fun(void *arg)
{
…
for(;;)
{
indi ++;
…
u0 = recv_msg->u0;
u1e = recv_msg->u1p;
u2e = recv_msg->u2p;
u3e = recv_msg->u3p;
u4e = recv_msg->u4p;
…
volts = u0;
if (indi = 5)                    //* 20% packet losses
{
volts =u1p;
indi = 0;
}
}
…
}

int main(void)
{
…
fprintf(fp, "%f\n", volts);
printf("Ball Position = %f mm\n", volts);
…
}
```

C. 3 C Codes for Various Command Inputs

C.3.1 Step Input

(1) Step Input

```
#include <stdio.h>
…
double y_hat_desi = 0.0004;       //* User input (desired set point)
double flag_sig = 0;
…

int main(void)
{
…
while(1)
{
flag_sig ++;
…
if (flag_sig >3000);    //*Step input starts at 10 s
{
y_hat_desi = -0.0006;
}
…
}
…
}
```

(2) Multiple Steps Input

```
#include <stdio.h>
```

```
…
double y_hat_desi = 0.0006;       //* User input (desired set point)
double flag_sig = 0;
…

int main(void)
{
…
while(1)
{
flag_sig ++;
…
if (flag_sig >3000);
{
y_hat_desi = 0.0003;
}
if (flag_sig >6000);
{
y_hat_desi = 0;
}
if (flag_sig >9000);
{
y_hat_desi = -0.0003;
}
if (flag_sig >3000);
{
y_hat_desi = -0.0006;
}
…
}
…
}
```

C.3.2 Sinusoidal Command Tracking

```
#include <stdio.h>
…
long indi = 0;
long flag_sig = 0;
double tt = 0;
double sindata[15709];
…
int main(void)
{
…
while(indi < 15709)    //* Sinusoidal command generation
{
sindata[indi]= -0.0002+0.0012*sin(tt);
tt = tt +0.0004;
indi ++;
}
while(1)
{
y_hat_desi = sindata[flag_sig];
flag_sig ++;
…
}
…
}
```

C.3.3 Saw-tooth Command Tracking

```
#include <stdio.h>
…
```

```
long flag_sig = 0;
…
int main(void)
{
…
while(1)
{
flag_sig ++;
…
if (flag_sig > 4000)
{
y_hat_desi = y_hat_desi - 0.0000008;
}
if(flag_sig > 5500)
{
y_hat_desi = y_hat_desi +0.0000016;
}
if (flag_sig >7000)
{
y_hat_desi = y_hat_desi -0.0000016;
}
if (flag_sig > 8500)
{
y_hat_desi = y_hat_desi + 0.0000016;
}
if (flag_sig > 10000)
{
y_hat_desi = y_hat_desi - 0.0000016;
}
if (flag_sig > 11500)
{
y_hat_desi = y_hat_desi + 0.0000016;
```

```
}
if (flag_sig > 13000)
{
y_hat_desi = y_hat_desi - 0.0000016;
}
if (flag_sig > 14500)
{
y_hat_desi = y_hat_desi + 0.0000016;
}
…
}
…
}
```

VITA

KUN JI

Education

Texas A&M University, College Station, Texas. May 2006,
Doctor of Philosophy in Mechanical Engineering.

Tsinghua University, Beijing, China. August 2002,
Master of Science in Mechanical Engineering.

Tsinghua University, Beijing, China. August 1999,
Bachelor of Science in Mechanical Engineering.

Journal Publications

[1] W.-J. Kim, **K, Ji**, and A. Ambike, "Networked real-time control strategy dealing with stochastic time delays and packet losses," *ASME Journal of Dynamic Systems, Measurement and Control*, in press.

[2] W.-J. Kim, **K. Ji**, and A. Ambike, "Real-time operating environment for networked control systems," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no.2, in press.

[3] **K. Ji**, W.-J. Kim, and A. Srivastava, "Internet-based real-time control architectures with time-delay/packet-loss compensation," *Asian Journal of Control*, vol. 9, no.1, in press.

[4] W.-J. Kim, **K. Ji**, and A. Srivastava, "Network-based control with real-time prediction of delayed/lost sensor data," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 1, pp. 182–185, Jan. 2006.

[5] **K. Ji** and W.-J. Kim, "Real-time control of networked control systems via Ethernet," *International Journal of Control, Automation, and Systems*, vol. 3, no. 4, pp. 591–600, Dec. 2005.

Conference Presentations

[1] **K. Ji** and W.-J. Kim, "Robust control for networked control systems with admissible parameter uncertainties," in *Proc. of 2005 ASME International Mechanical Engineering Congress and Exposition*. Paper No. 81551, Nov. 2005.

[2] W.-J. Kim, **K. Ji**, and A. Ambike, "Networked real-time control strategies dealing with stochastic time delays and packet losses," in *Proc. of 2005 American Control Conference*, pp. 621–626, June 2005.

[3] A. Ambike., W.-J. Kim, and **K. Ji**, "Real-time operating environment for networked control systems," in *Proc. of 2005 American Control Conference*, pp. 2353–2358, June 2005.

[4] **K. Ji**, W.-J. Kim, and A. Ambike, "Control strategies for distributed real-time control with time delays and packets losses," in *Proc. of 2004 ASME International Mechanical Engineering Congress and Exposition*, Paper No. 61733, Nov. 2004.